Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	ShadowFi Protocol
Website:	https://shadowfi.com
Platform:	Binance Smart Chain
Language:	Solidity
Date:	August 25th, 2022

Table of contents

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 THE PUBLIC AFTER ISSUES ARE RESOLVED.

Introduction

EtherAuthority was contracted by ShadowFi to perform the Security audit of the ShadowFi 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 August 25th, 2022.

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

- ShadowFi believes in the protection of personally identifiable information from corporate and global financial entities.
- ShadowFi is a smart contract which has functions like withdraw, endLock, extendLockTime, withdrawBNB, withdrawTokens, shadowStrike, shadowBurst, receive, addLiquidity, burn, airdrop, addPair, buy, etc.
- The ShadowFi contract inherits the ReentrancyGuard. standard smart contracts from the OpenZeppelin library. These OpenZeppelin contracts are considered community-audited and time-tested, and hence are not part of the audit scope.

Audit scope

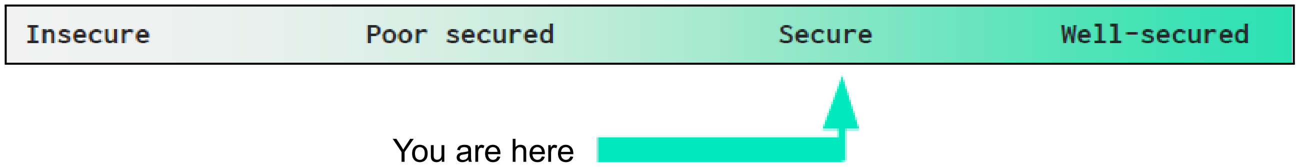
Name	**Code Review and Security Analysis Report for ShadowFi Protocol Smart Contracts**
Platform	**BSC / Solidity**
File 1	ShadowFiLPVault.sol
File 1 MD5 Hash	BA68AAC1F4C3FEA3EE2AD2F39DE6AD23
File 2	ShadowFiPresale.sol
File 2 MD5 Hash	00EAAB43804D28E63073D45909647B2C
File 3	ShadowFiToken.sol
File 3 MD5 Hash	DFBFABC899447D8205E7C0912C4969F7
Audit Date	August 25th,2022

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 ShadowFiLPVault.sol ● Owner can set the end lock time. ● Owner can withdraw BNB tokens. ● Owner can add a new Liquidity pool.	**YES, This is valid.**
File 2 ShadowFiPresale.sol ● Owner can deposit tokens, withdraw tokens. ● Owner can set the cost price, tokens, and discount percentage.	**YES, This is valid.**
File 3 ShadowFiToken.sol ● Name: ShadowFi ● Symbol: SDF ● Decimals: 9 ● Total Supply: ● Maximum Supply: ● Maximum Tax Amount: 0.1% ● Liquidity Fee: 2% ● Reflection Fee: 6% ● MarketingFee: 1% ● Total Buy Fee: 9% ● Total Sell Fee: 14% ● Fee Denominator: 10000 ● Target Liquidity Denominator: 100 ● Target Liquidity: 20 ● Buyback Multiplier Numerator: 150 ● Buyback Multiplier Denominator: 100 ● Buyback Multiplier Length: 30 minutes ● Distributor Gas: 500000 ● Swap Threshold: 0.02%	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **“ Secured”**. Also, these contracts do contain owner control, which does not make them fully decentralized.



We used various tools like Slither, Solhint 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 and 1 low and some very low level issues.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	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	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Moderated
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	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 3 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in ShadowFi 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 ShadowFi.

The ShadowFi team has provided unit test scripts, which would have helped to determine the integrity of the code in an automated way.

All code parts are well commented on smart contracts.

Documentation

We were given a ShadowFi smart contract code in the form of a file. The hash of that code is mentioned above in the table.

As mentioned above, code parts are well commented. And the logic is straightforward. So it is easy 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://shadowfi.com which provided rich information about the project architecture.

Use of Dependencies

As per our observation, the libraries are used in this smart contracts infrastructure that are based on well known industry standard open source projects.

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

AS-IS overview

ShadowFiLPVault.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	transferOwnership	write	access only Owner	No Issue
5	_transferOwnership	internal	Passed	No Issue
6	endLock	write	access only Owner	No Issue
7	extendLockTime	write	access only Owner	No Issue
8	withdrawBNB	write	access only Owner	No Issue
9	withdrawTokens	write	Passed	No Issue
10	shadowStrike	write	access only Owner	No Issue
11	shadowBurst	write	access only Owner	No Issue
12	getLPOwnershipPercent	read	Passed	No Issue
13	getLiquidTokens	read	Passed	No Issue
14	getLiquidPercent	read	Passed	No Issue
15	getRemoveAmountForBuyAndBurnExcess	read	Passed	No Issue
16	addLiquidity	external	access only Owner	No Issue
17	getLockTime	read	Passed	No Issue
18	receive	external	Passed	No Issue
19	fallback	external	Passed	No Issue
20	_nonReentrantBefore	modifier	Passed	No Issue
21	_nonReentrantBefore	write	Passed	No Issue
22	_nonReentrantAfter	write	Passed	No Issue

ShadowFiPresale.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Remove unnecessary code	Refer Audit Findings
2	_nonReentrantBefore	modifier	Passed	No Issue
3	_nonReentrantBefore	write	Passed	No Issue
4	_nonReentrantAfter	write	Passed	No Issue
5	owner	read	Passed	No Issue
6	onlyOwner	modifier	Passed	No Issue
7	transferOwnership	write	access only Owner	No Issue
8	_transferOwnership	internal	Passed	No Issue
9	depositTokens	write	access only Owner	No Issue
10	withdrawTokens	write	access only Owner	No Issue

11	setCost	write	access only Owner	No Issue
12	setToken	write	access only Owner	No Issue
13	setDiscount	write	access only Owner	No Issue
14	setStartandStopTime	write	access only Owner	No Issue
15	setMax	write	access only Owner	No Issue
16	withdraw	write	access only Owner	No Issue
17	tokenAddress	read	Passed	No Issue
18	availableForSaleTokenAmount	read	Passed	No Issue
19	totalBoughtByUserTokenAmount	read	Passed	No Issue
20	totalSoldTokenAmount	read	Passed	No Issue
21	tokenCostBNB	read	Passed	No Issue
22	totalBNBRaisedSoFar	read	Passed	No Issue
23	discountPercentage	read	Passed	No Issue
24	maxBuyableTokenAmount	read	Passed	No Issue
25	getStartTime	read	Passed	No Issue
26	getStopTime	read	Passed	No Issue
27	buy	external	Passed	No Issue

ShadowFiToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	authorizedFor	modifier	Passed	No Issue
4	authorizeFor	write	access by authorized For	No Issue
5	authorizeForMultiplePermissions	write	access by authorized For	No Issue
6	unauthorizeFor	write	access by authorized For	No Issue
7	unauthorizeForMultiplePermissions	write	access by authorized For	No Issue
8	isOwner	read	Passed	No Issue
9	isAuthorizedFor	read	Passed	No Issue
10	isAuthorizedFor	read	Passed	No Issue
11	transferOwnership	write	access only Owner	No Issue
12	getPermissionNameToIndex	read	Passed	No Issue
13	getPermissionUnlockTime	read	Passed	No Issue
14	isLocked	read	Passed	No Issue
15	lockPermission	write	Missing required check condition	Refer Audit Findings
16	unlockPermission	write	Missing required check condition	Refer Audit Findings
17	receive	external	Passed	No Issue

18	totalSupply	external	Passed	No Issue
19	decimals	external	Passed	No Issue
20	symbol	external	Passed	No Issue
21	name	external	Passed	No Issue
22	getOwner	external	Passed	No Issue
23	balanceOf	read	Passed	No Issue
24	allowance	external	Passed	No Issue
25	approve	write	Passed	No Issue
26	approveMax	external	Passed	No Issue
27	transfer	external	Passed	No Issue
28	transferFrom	external	Passed	No Issue
29	_transferFrom	internal	Passed	No Issue
30	_basicTransfer	internal	Passed	No Issue
31	checkTxLimit	internal	Passed	No Issue
32	shouldTakeFee	internal	Passed	No Issue
33	getTotalFee	read	Passed	No Issue
34	getAdditionalTaxFee	read	Passed	No Issue
35	getMultipliedFee	read	Passed	No Issue
36	takeFee	internal	Passed	No Issue
37	isSell	internal	Passed	No Issue
38	shouldSwapBack	internal	Passed	No Issue
39	swapBack	internal	Add Liquidity with External account	Refer Audit Findings
40	triggerBuyback	external	access by authorized For	No Issue
41	clearBuybackMultiplier	external	access by authorized For	No Issue
42	buyTokens	internal	access by authorized For	No Issue
43	setBuybackMultiplierSettings	external	access by authorized For	No Issue
44	launched	internal	Passed	No Issue
45	launch	internal	Passed	No Issue
46	setTxLimit	external	access by authorized For	No Issue
47	setIsDividendExempt	external	access by authorized For	No Issue
48	setIsFeeExempt	external	access by authorized For	No Issue
49	setIsTxLimitExempt	external	access by authorized For	No Issue
50	setFees	external	access by authorized For	No Issue
51	setFeeReceivers	external	access by authorized For	No Issue
52	setSwapBackSettings	external	access by authorized For	No Issue

53	setTargetLiquidity	external	access by authorized For	No Issue
54	setDistributionCriteria	external	access by authorized For	No Issue
55	setDistributorSettings	external	access by authorized For	No Issue
56	getCirculatingSupply	read	Passed	No Issue
57	getLiquidityBacking	read	Passed	No Issue
58	isOverLiquified	read	Passed	No Issue
59	claimDividend	external	Missing required check condition	Refer Audit Findings
60	addPair	external	access by authorized For	No Issue
61	removeLastPair	external	access by authorized For	No Issue
62	setFeesOnNormalTransfers	external	access by authorized For	No Issue
63	setLaunchedAt	external	access by authorized For	No Issue
64	setAllowedAddress	external	access only Owner	No Issue
65	burn	write	Passed	No Issue
66	airdrop	external	access only Owner	No Issue
67	isAirdropped	external	Passed	No Issue
68	setBlackListed	external	access only Owner	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
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens loss
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 Severity

No Critical severity vulnerabilities were found.

High Severity

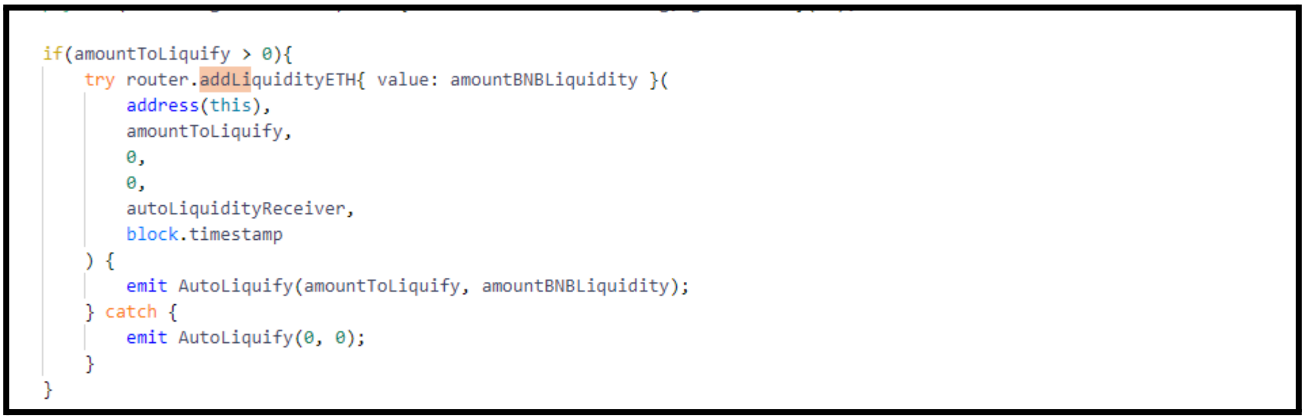
No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Add Liquidity with External account:**- ShadowFiToken.sol**



```
if(amountToLiquify > 0){
    try router.addLiquidityETH{ value: amountBNBLiquidity }(
        address(this),
        amountToLiquify,
        0,
        0,
        autoLiquidityReceiver,
        block.timestamp
    ) {
        emit AutoLiquify(amountToLiquify, amountBNBLiquidity);
    } catch {
        emit AutoLiquify(0, 0);
    }
}
```

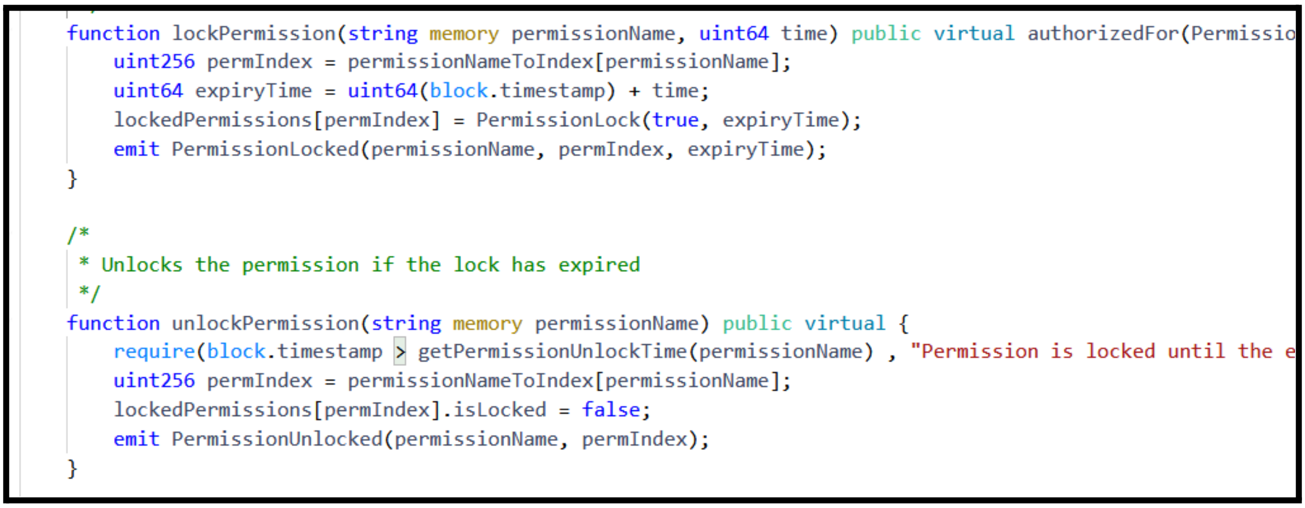
addLiquidity function of pancakeswapRouter with the address specified as autoLiquidityReceiver for acquiring the SDF tokens. As a result, over time the autoLiquidityReceiver address will accumulate a significant portion of SDF tokens. If the autoLiquidityReceiver is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Resolution: We advise the address of the addLiquidity function call to be replaced by the contract itself, i.e. address(this), and to restrict the management of the SDF tokens within the scope of the contract's business logic. This will also protect the SDF tokens from being stolen if the autoLiquidityReceiver account is compromised.

Very Low / Informational / Best practices:

(1) Missing required check condition:

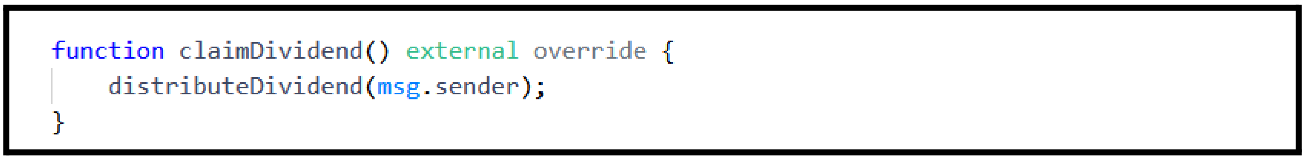
ShadowFiToken.sol



```
    function lockPermission(string memory permissionName, uint64 time) public virtual authorizedFor(Permissio
        uint256 permIndex = permissionNameToIndex[permissionName];
        uint64 expiryTime = uint64(block.timestamp) + time;
        lockedPermissions[permIndex] = PermissionLock(true, expiryTime);
        emit PermissionLocked(permissionName, permIndex, expiryTime);
    }

    /*
     * Unlocks the permission if the lock has expired
     */
    function unlockPermission(string memory permissionName) public virtual {
        require(block.timestamp > getPermissionUnlockTime(permissionName) , "Permission is locked until the e
        uint256 permIndex = permissionNameToIndex[permissionName];
        lockedPermissions[permIndex].isLocked = false;
        emit PermissionUnlocked(permissionName, permIndex);
    }
```

There should be a check for lockPermission or unlockPermission before locking or unlocking.

Resolution: There should be a check to identify if the state is locked, then only we can unlock and vice versa.

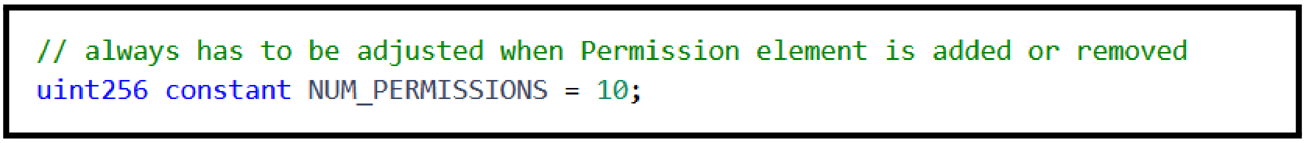


```
function claimDividend() external override {
    distributeDividend(msg.sender);
}
```

Please use a required check for a valid shareholder check, this will save on unnecessary gas fees.

Resolution: We suggest using a record to check valid shareholders.

(2) Irrelevant comment:- **ShadowFiToken.sol**

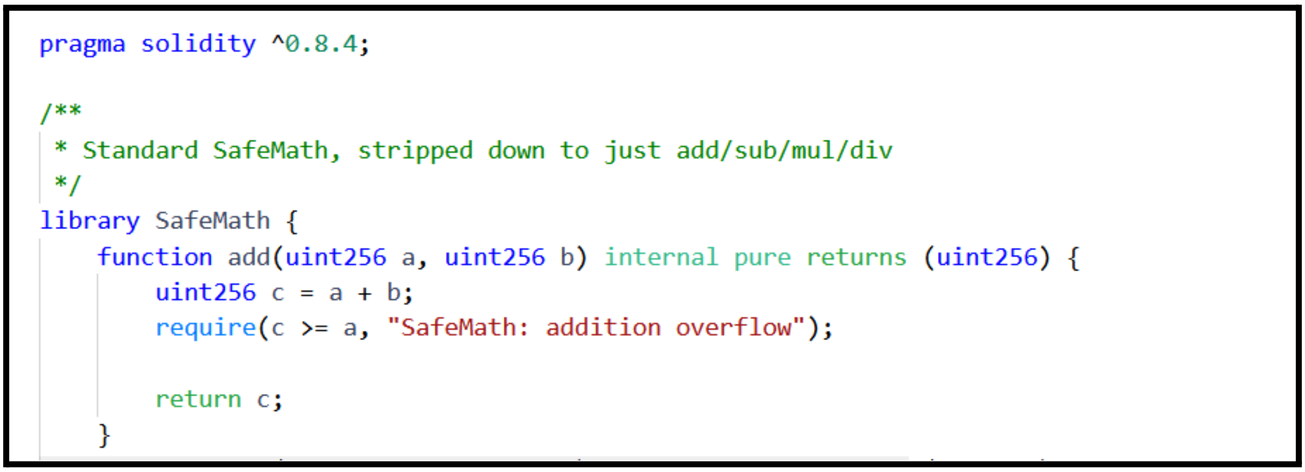


```
// always has to be adjusted when Permission element is added or removed
uint256 constant NUM_PERMISSIONS = 10;
```

There is a comment where you mention "NUM_PERMISSIONS" will adjust when the Permission element is added or removed, but it can't because it is a constant variable.

Resolution: We suggest updating your comment.

(3) SafeMath library avoid :- **ShadowFiToken.sol**

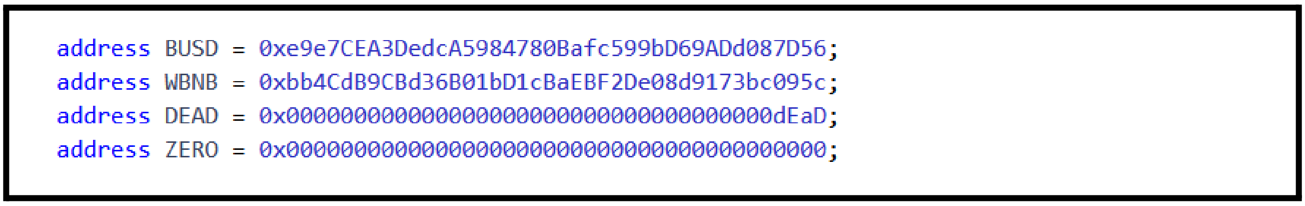


```
pragma solidity ^0.8.4;

/**
 * Standard SafeMath, stripped down to just add/sub/mul/div
 */
library SafeMath {
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");

        return c;
    }
```

After the release of solidity 0.8.0 or greater "safeMath" is not mandatory.

Resolution: We suggest avoiding use of SafeMath.

(4) Declared hard coded variables as constant:- **ShadowFiToken.sol**

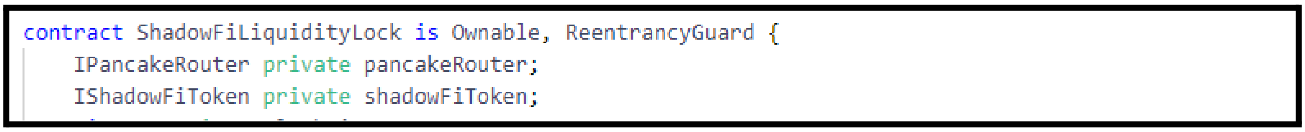


```
  address BUSD = 0xe9e7CEA3DedcA5984780Bafc599bD69ADd087D56;
  address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c;
  address DEAD = 0x000000000000000000000000000000000000dEaD;
  address ZERO = 0x0000000000000000000000000000000000000000;
```

Some variables are set as hard coded addresses and never reset.

Resolution: Deployer has to confirm before deploying the contract to production and declare them as constant.

(5) Immutable variables: **ShadowFiLPVault.sol**

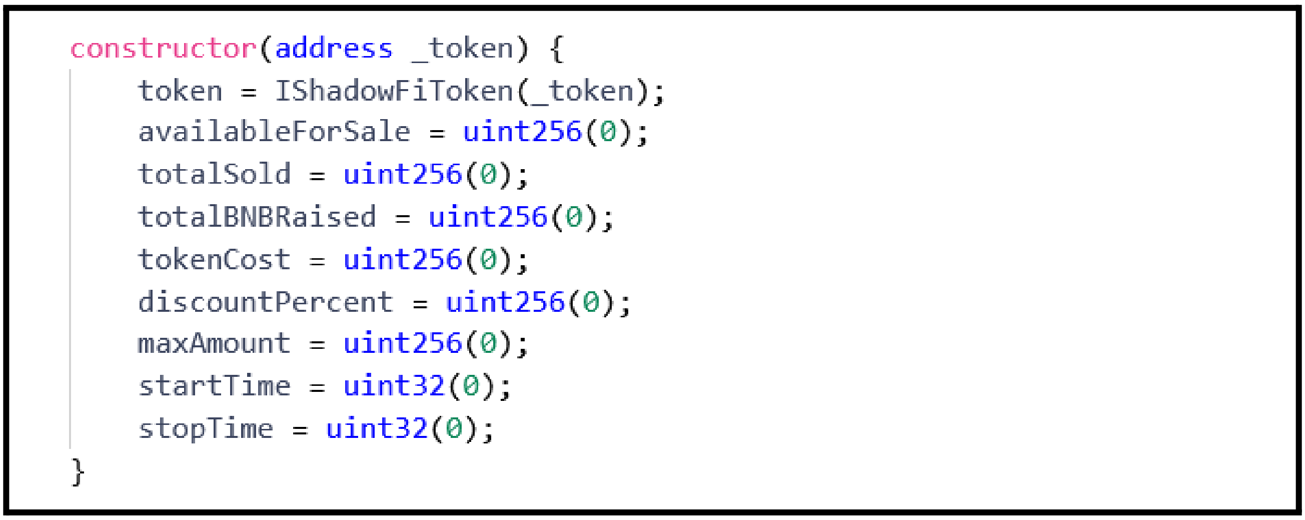


```
contract ShadowFiLiquidityLock is Ownable, ReentrancyGuard {
    IPancakeRouter private pancakeRouter;
    IShadowFiToken private shadowFiToken;
```

Variables that are defined within the constructor but further remain unchanged should be marked as immutable to save gas and to ease the reviewing process of third-parties.

Resolution: Consider marking this variable as immutable.

(6) Variables initialized by default value:- **ShadowFiPresale.sol**



```
constructor(address _token) {
    token = IShadowFiToken(_token);
    availableForSale = uint256(0);
    totalSold = uint256(0);
    totalBNBRaised = uint256(0);
    tokenCost = uint256(0);
    discountPercent = uint256(0);
    maxAmount = uint256(0);
    startTime = uint32(0);
    stopTime = uint32(0);
}
```

Solidity uses 0 as the default value of int variables, so there is no need to reassign the 0 values.

Resolution: We suggest ignoring to initialize the variables by default value.

Centralization

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

- endLock: ShadowFiLPVault owner can set end lock.
- extendLockTime: ShadowFiLPVault owner can extend lock time.
- withdrawBNB: ShadowFiLPVault owner can withdraw BNB token.
- withdrawTokens: ShadowFiLPVault owner can withdraw tokens.
- shadowStrike: ShadowFiLPVault owner can shadow strike.
- shadowBurst: ShadowFiLPVault owner can shadow burst value.
- addLiquidity: ShadowFiLPVault owner can add new liquidity.
- depositTokens: ShadowFiPresale owner can deposit tokes.
- withdrawTokens: ShadowFiPresale owner can withdraw tokens.
- setCost: ShadowFiPresale owner can set cost.
- setToken: ShadowFiPresale owner can set tokens.

- setDiscount: ShadowFiPresale owner can set discount percentage.
- setStartandStopTime: ShadowFiPresale owner can set start and stop time.
- setMax: ShadowFiPresale owner can set maximum amount.
- withdraw: ShadowFiPresale owner can withdraw amount.
- triggerBuyback: ShadowFiToken owner can trigger buy back value.
- clearBuybackMultiplier: ShadowFiToken authorized owner can clear buyback multiplier.
- setBuybackMultiplierSettings: ShadowFiToken authorized owner can set buyback multiplier settings.
- setTxLimit: ShadowFiToken authorized owner can transfer limit.
- setIsDividendExempt: ShadowFiToken owner can set its dividend exempt.
- setIsFeeExempt: ShadowFiToken authorized owner can set fee exempt.
- setIsTxLimitExempt: ShadowFiToken authorized owner can set transfer limit. exempt.
- setFees: ShadowFiToken authorized owner can set liquidity Fee, buyback Fee, reflection Fee, marketing Fee, fee Denominator, total Sell Fee.
- setFeeReceivers: ShadowFiToken owner can set receivers fee.
- setSwapBackSettings: ShadowFiToken authorize owner can set swap back settings.
- setTargetLiquidity: ShadowFiToken authorized owner can set target liquidity.
- setDistributionCriteria: ShadowFiToken authorize owner can set distribution criteria.
- setDistributorSettings: ShadowFiToken authorize owner can set distribution settings.
- addPair: ShadowFiToken authorize owner can add pair address.
- removeLastPair: ShadowFiToken authorized owner can remove last pair address.
- setFeesOnNormalTransfers: ShadowFiToken authorize owner can set fees on normal transfers status.
- setLaunchedAt: ShadowFiToken authorize owner can set launched value.
- setAllowedAddress: ShadowFiToken owner can set allowed address.
- airdrop: ShadowFiToken owner can airdrop tokens.
- setBlackListed: ShadowFiToken owner can set black listed addresses.

To make the smart contract 100% decentralized, we suggest renouncing ownership in the smart contract once its function is completed.

Conclusion

We were given a contract code in the form of a file. And we have used all possible tests based on given objects as files. We have observed 1 Low severity issue in smart contracts. But that is not a critical one. **So, the smart contracts are ready for the mainnet deployment**.

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 the 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 **"Secure".**

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.

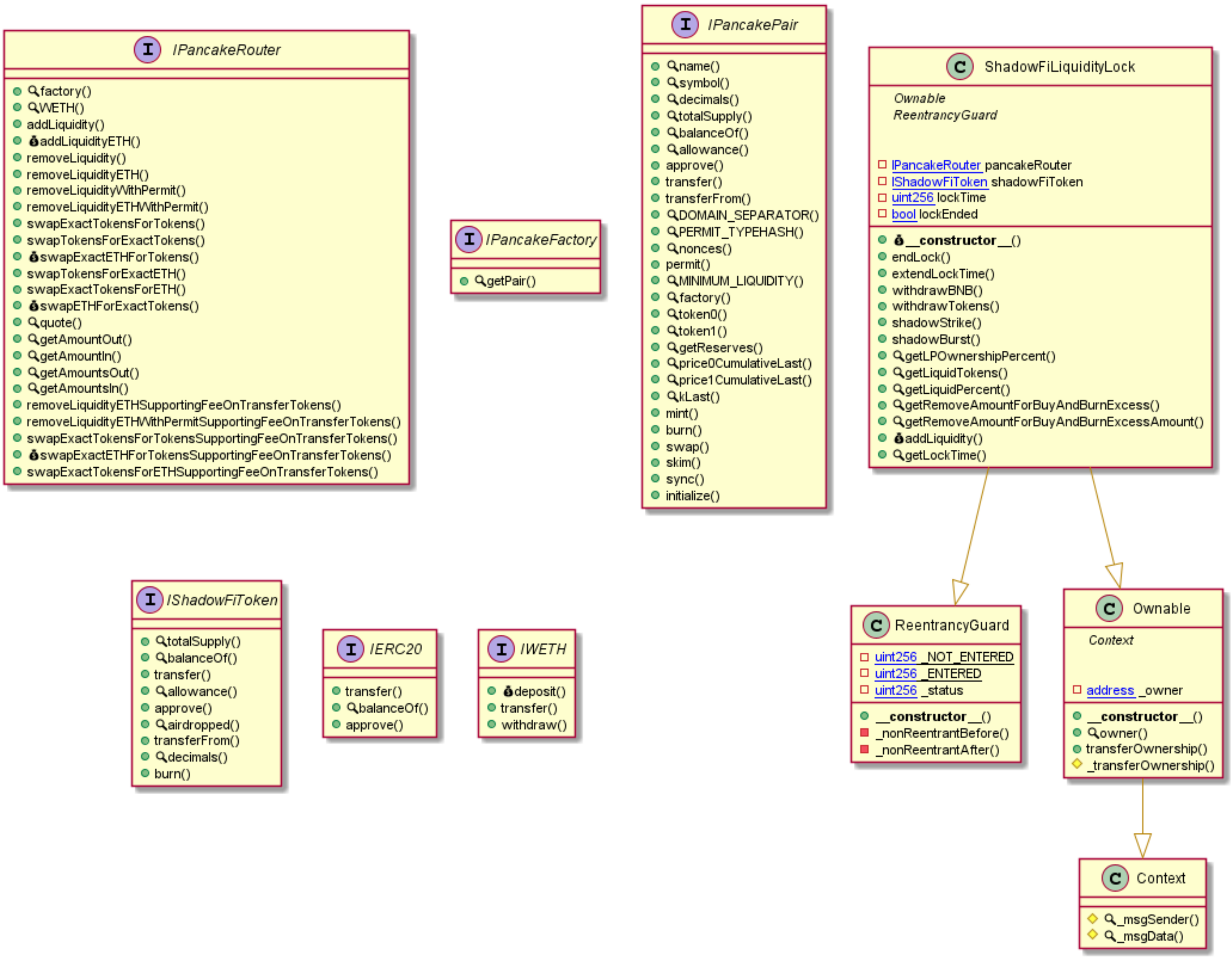
Technical Disclaimer

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.

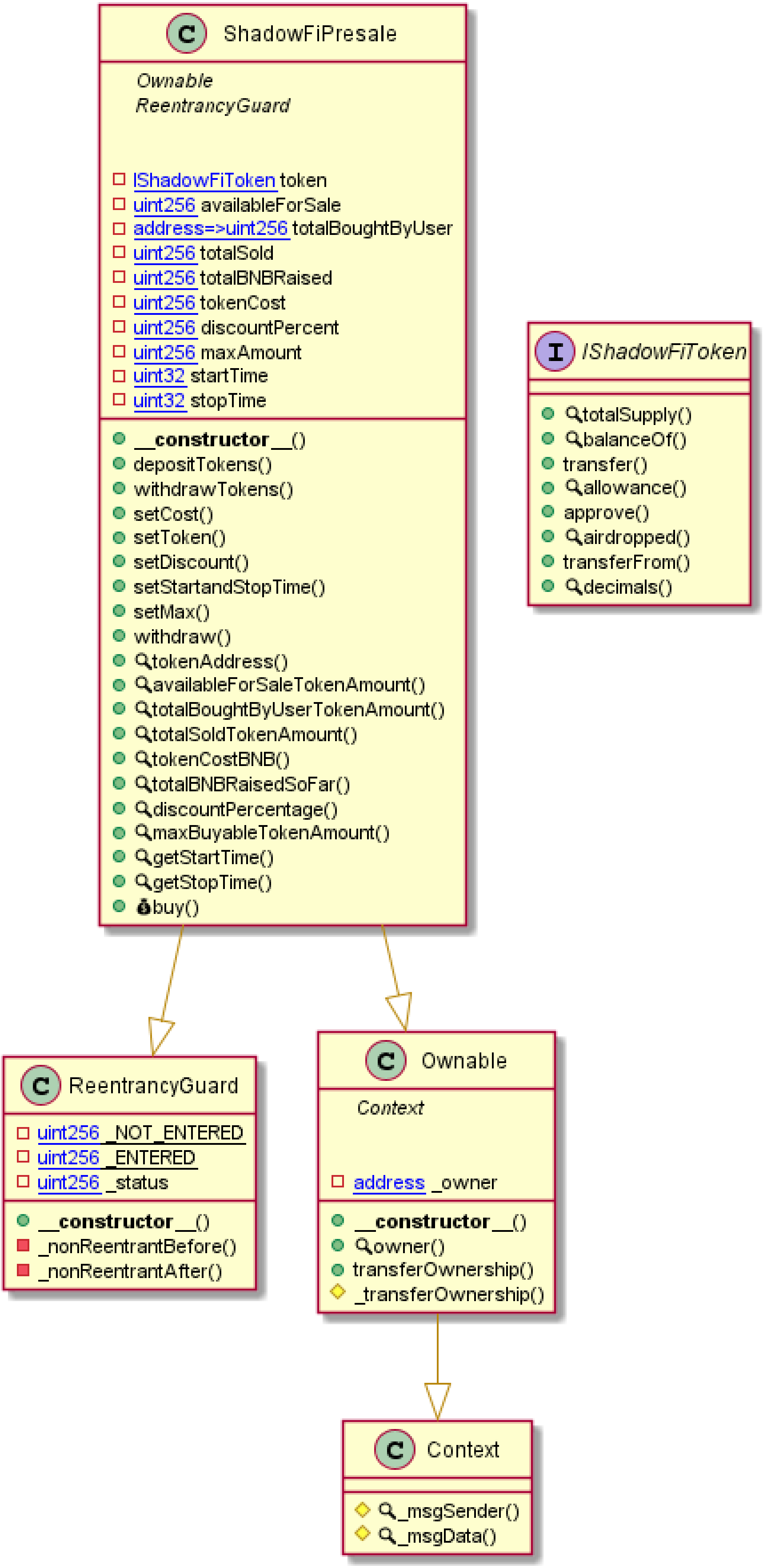
Appendix

Code Flow Diagram - ShadowFi Protocol

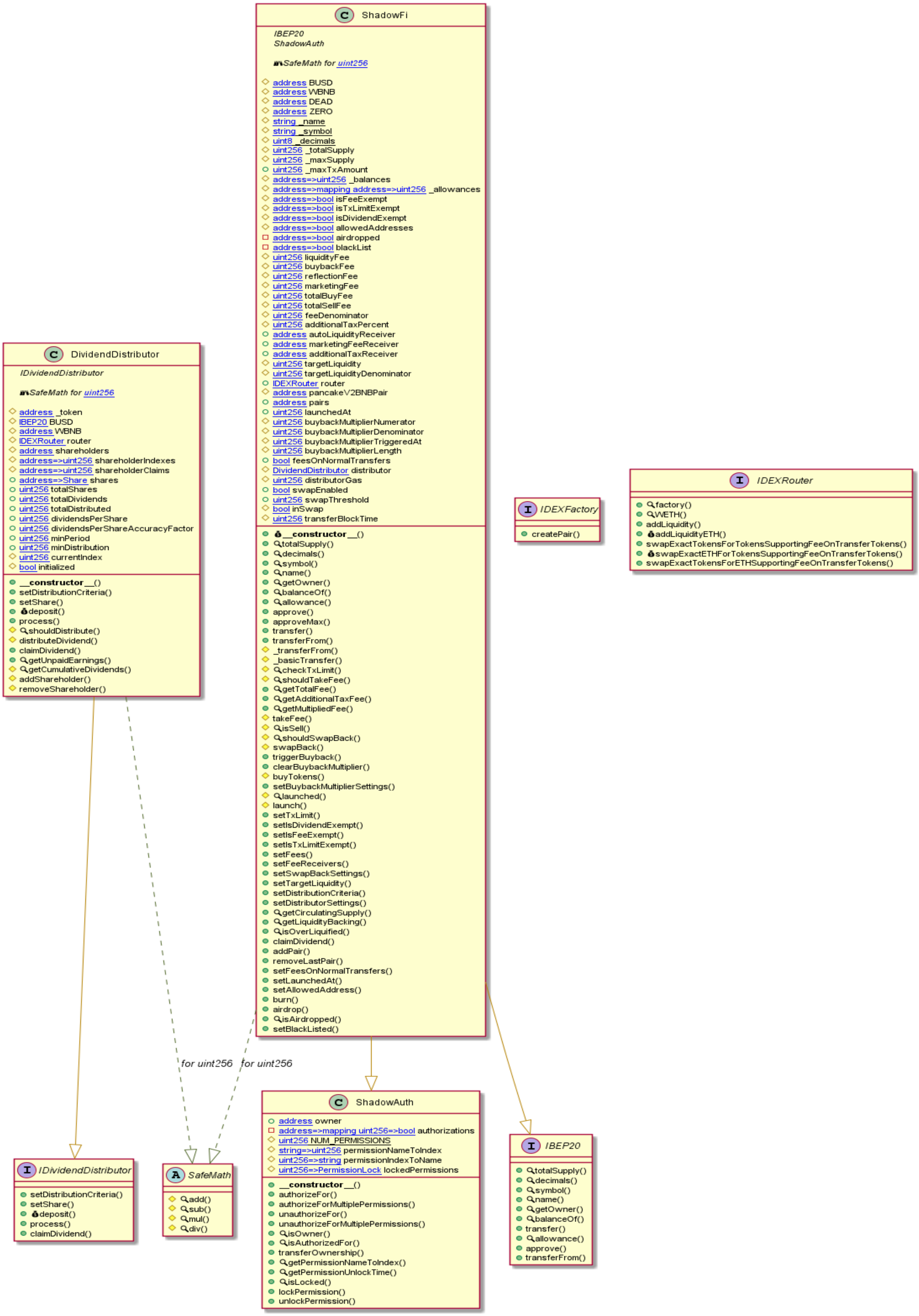
ShadowFiLPVault Diagram



ShadowFiPresale Diagram

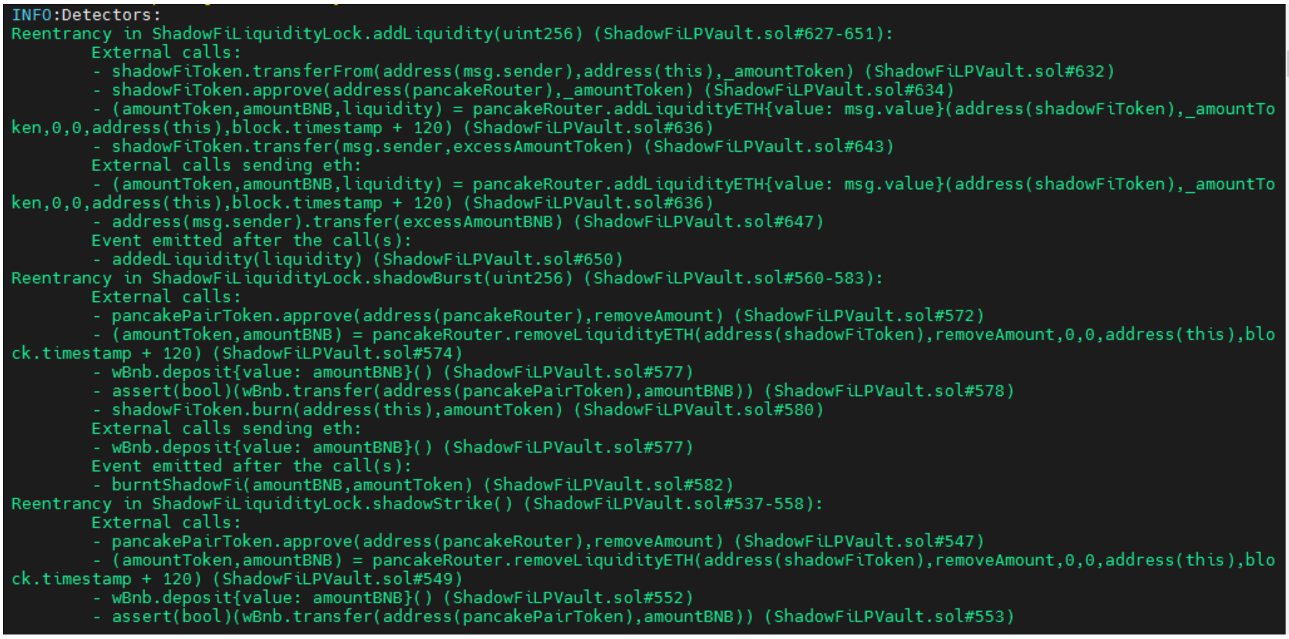


ShadowFiToken Diagram

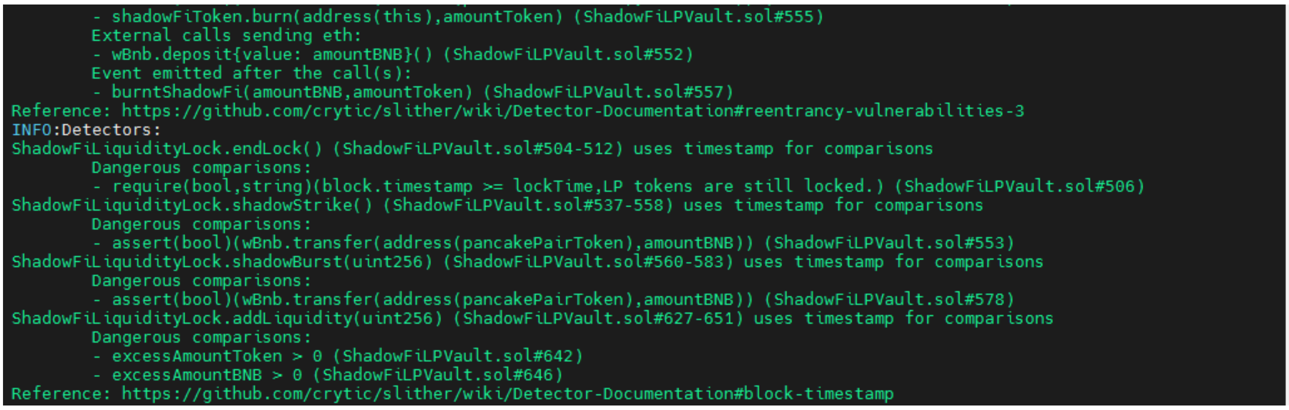


Slither Results Log

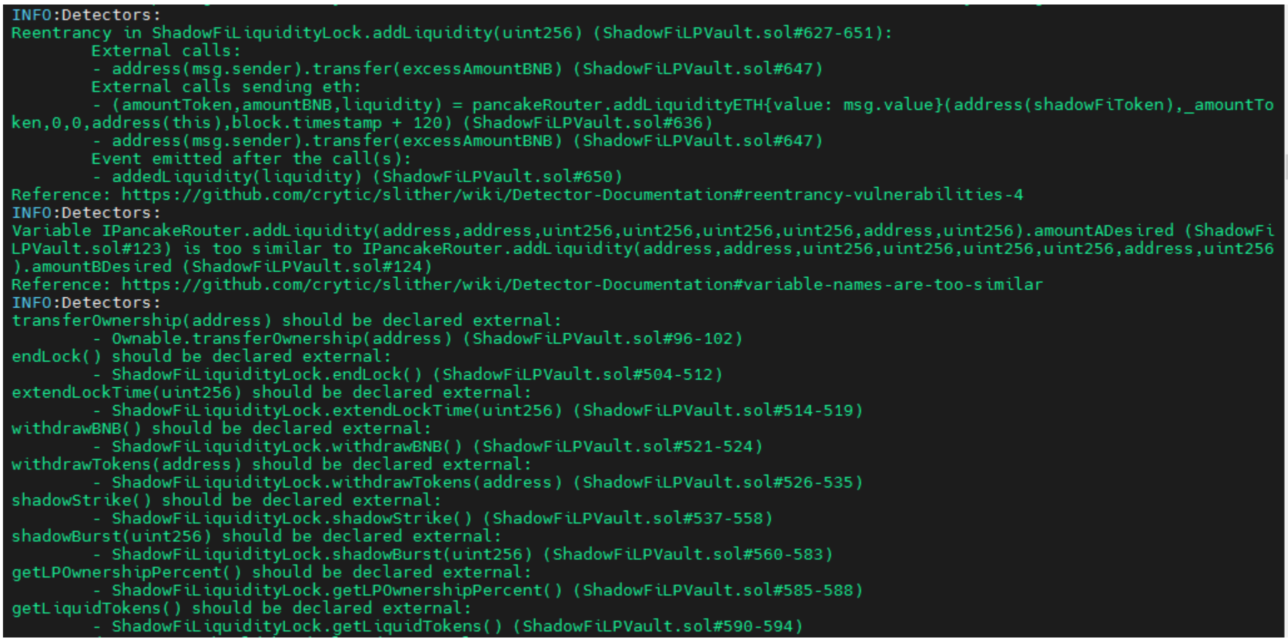
Slither log >> ShadowFiLPVault.sol



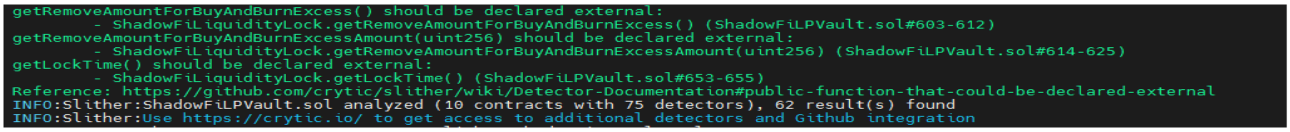
```
INFO:Detectors:
Reentrancy in ShadowFiLiquidityLock.addLiquidity(uint256) (ShadowFiLPVault.sol#627-651):
        External calls:
        - shadowFiToken.transferFrom(address(msg.sender),address(this),_amountToken) (ShadowFiLPVault.sol#632)
        - shadowFiToken.approve(address(pancakeRouter),_amountToken) (ShadowFiLPVault.sol#634)
        - (amountToken,amountBNB,liquidity) = pancakeRouter.addLiquidityETH{value: msg.value}(address(shadowFiToken),_amountTo
ken,0,0,address(this),block.timestamp + 120) (ShadowFiLPVault.sol#636)
        - shadowFiToken.transfer(msg.sender,excessAmountToken) (ShadowFiLPVault.sol#643)
        External calls sending eth:
        - (amountToken,amountBNB,liquidity) = pancakeRouter.addLiquidityETH{value: msg.value}(address(shadowFiToken),_amountTo
ken,0,0,address(this),block.timestamp + 120) (ShadowFiLPVault.sol#636)
        - address(msg.sender).transfer(excessAmountBNB) (ShadowFiLPVault.sol#647)
        Event emitted after the call(s):
        - addedLiquidity(liquidity) (ShadowFiLPVault.sol#650)
Reentrancy in ShadowFiLiquidityLock.shadowBurst(uint256) (ShadowFiLPVault.sol#560-583):
        External calls:
        - pancakePairToken.approve(address(pancakeRouter),removeAmount) (ShadowFiLPVault.sol#572)
        - (amountToken,amountBNB) = pancakeRouter.removeLiquidityETH(address(shadowFiToken),removeAmount,0,0,address(this),blo
ck.timestamp + 120) (ShadowFiLPVault.sol#574)
        - wBnb.deposit{value: amountBNB}() (ShadowFiLPVault.sol#577)
        - assert(bool)(wBnb.transfer(address(pancakePairToken),amountBNB)) (ShadowFiLPVault.sol#578)
        - shadowFiToken.burn(address(this),amountToken) (ShadowFiLPVault.sol#580)
        External calls sending eth:
        - wBnb.deposit{value: amountBNB}() (ShadowFiLPVault.sol#577)
        Event emitted after the call(s):
        - burntShadowFi(amountBNB,amountToken) (ShadowFiLPVault.sol#582)
Reentrancy in ShadowFiLiquidityLock.shadowStrike() (ShadowFiLPVault.sol#537-558):
        External calls:
        - pancakePairToken.approve(address(pancakeRouter),removeAmount) (ShadowFiLPVault.sol#547)
        - (amountToken,amountBNB) = pancakeRouter.removeLiquidityETH(address(shadowFiToken),removeAmount,0,0,address(this),blo
ck.timestamp + 120) (ShadowFiLPVault.sol#549)
        - wBnb.deposit{value: amountBNB}() (ShadowFiLPVault.sol#552)
        - assert(bool)(wBnb.transfer(address(pancakePairToken),amountBNB)) (ShadowFiLPVault.sol#553)
```



```
        - shadowFiToken.burn(address(this),amountToken) (ShadowFiLPVault.sol#555)
        External calls sending eth:
        - wBnb.deposit{value: amountBNB}() (ShadowFiLPVault.sol#552)
        Event emitted after the call(s):
        - burntShadowFi(amountBNB,amountToken) (ShadowFiLPVault.sol#557)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
ShadowFiLiquidityLock.endLock() (ShadowFiLPVault.sol#504-512) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= lockTime,LP tokens are still locked.) (ShadowFiLPVault.sol#506)
ShadowFiLiquidityLock.shadowStrike() (ShadowFiLPVault.sol#537-558) uses timestamp for comparisons
        Dangerous comparisons:
        - assert(bool)(wBnb.transfer(address(pancakePairToken),amountBNB)) (ShadowFiLPVault.sol#553)
ShadowFiLiquidityLock.shadowBurst(uint256) (ShadowFiLPVault.sol#560-583) uses timestamp for comparisons
        Dangerous comparisons:
        - assert(bool)(wBnb.transfer(address(pancakePairToken),amountBNB)) (ShadowFiLPVault.sol#578)
ShadowFiLiquidityLock.addLiquidity(uint256) (ShadowFiLPVault.sol#627-651) uses timestamp for comparisons
        Dangerous comparisons:
        - excessAmountToken > 0 (ShadowFiLPVault.sol#642)
        - excessAmountBNB > 0 (ShadowFiLPVault.sol#646)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

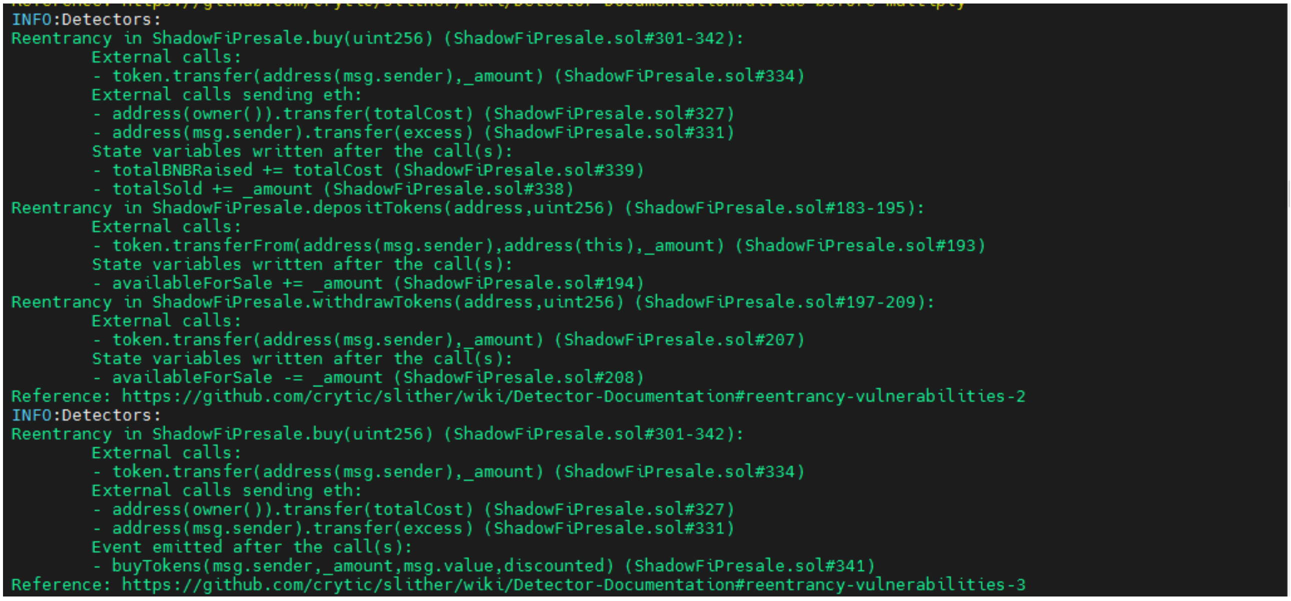


```
INFO:Detectors:
Reentrancy in ShadowFiLiquidityLock.addLiquidity(uint256) (ShadowFiLPVault.sol#627-651):
        External calls:
        - address(msg.sender).transfer(excessAmountBNB) (ShadowFiLPVault.sol#647)
        External calls sending eth:
        - (amountToken,amountBNB,liquidity) = pancakeRouter.addLiquidityETH{value: msg.value}(address(shadowFiToken),_amountTo
ken,0,0,address(this),block.timestamp + 120) (ShadowFiLPVault.sol#636)
        - address(msg.sender).transfer(excessAmountBNB) (ShadowFiLPVault.sol#647)
        Event emitted after the call(s):
        - addedLiquidity(liquidity) (ShadowFiLPVault.sol#650)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
Variable IPancakeRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (ShadowFi
LPVault.sol#123) is too similar to IPancakeRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256
).amountBDesired (ShadowFiLPVault.sol#124)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (ShadowFiLPVault.sol#96-102)
endLock() should be declared external:
        - ShadowFiLiquidityLock.endLock() (ShadowFiLPVault.sol#504-512)
extendLockTime(uint256) should be declared external:
        - ShadowFiLiquidityLock.extendLockTime(uint256) (ShadowFiLPVault.sol#514-519)
withdrawBNB() should be declared external:
        - ShadowFiLiquidityLock.withdrawBNB() (ShadowFiLPVault.sol#521-524)
withdrawTokens(address) should be declared external:
        - ShadowFiLiquidityLock.withdrawTokens(address) (ShadowFiLPVault.sol#526-535)
shadowStrike() should be declared external:
        - ShadowFiLiquidityLock.shadowStrike() (ShadowFiLPVault.sol#537-558)
shadowBurst(uint256) should be declared external:
        - ShadowFiLiquidityLock.shadowBurst(uint256) (ShadowFiLPVault.sol#560-583)
getLPOwnershipPercent() should be declared external:
        - ShadowFiLiquidityLock.getLPOwnershipPercent() (ShadowFiLPVault.sol#585-588)
getLiquidTokens() should be declared external:
        - ShadowFiLiquidityLock.getLiquidTokens() (ShadowFiLPVault.sol#590-594)
```

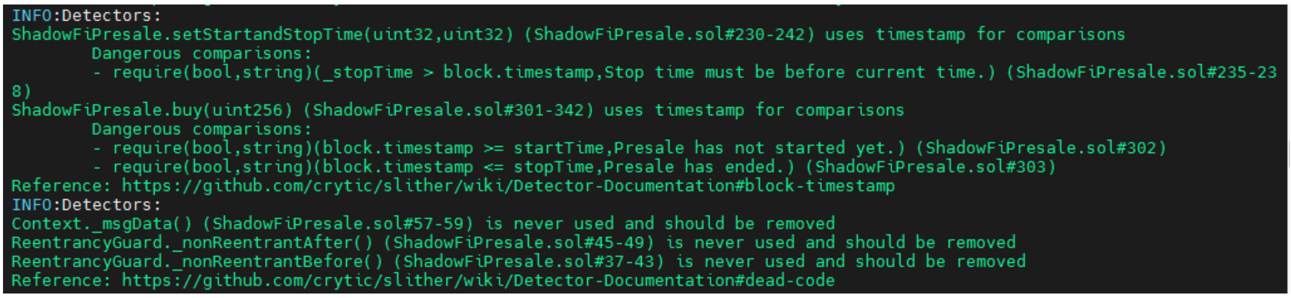


```
getRemoveAmountForBuyAndBurnExcess() should be declared external:
        - ShadowFiLiquidityLock.getRemoveAmountForBuyAndBurnExcess() (ShadowFiLPVault.sol#603-612)
getRemoveAmountForBuyAndBurnExcessAmount(uint256) should be declared external:
        - ShadowFiLiquidityLock.getRemoveAmountForBuyAndBurnExcessAmount(uint256) (ShadowFiLPVault.sol#614-625)
getLockTime() should be declared external:
        - ShadowFiLiquidityLock.getLockTime() (ShadowFiLPVault.sol#653-655)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ShadowFiLPVault.sol analyzed (10 contracts with 75 detectors), 62 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

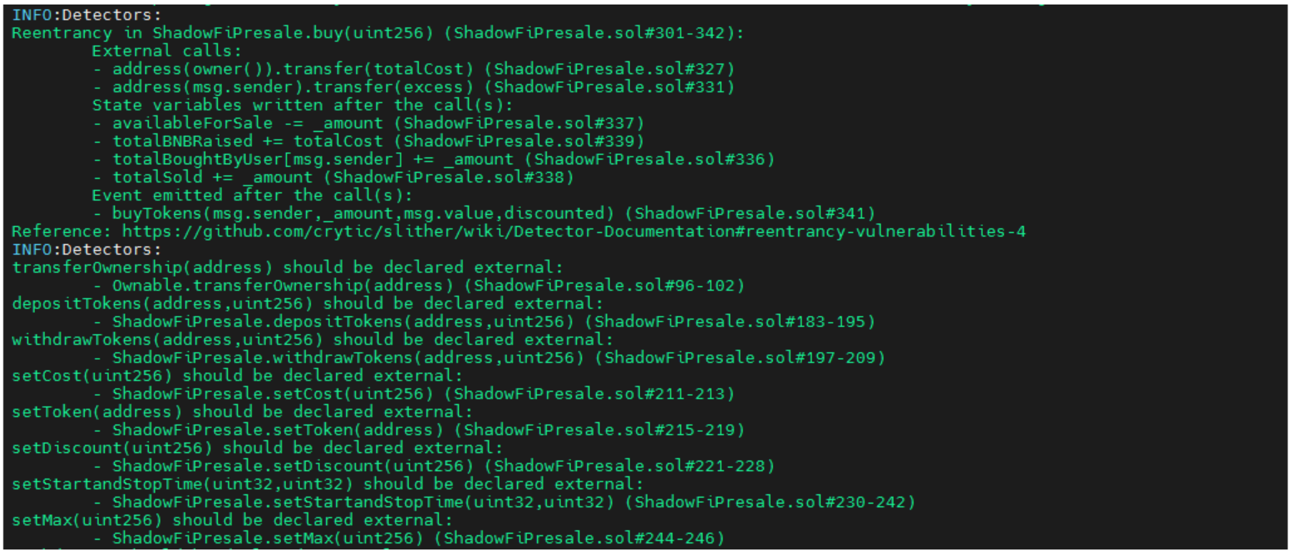
Slither log >> ShadowFiPresale.sol



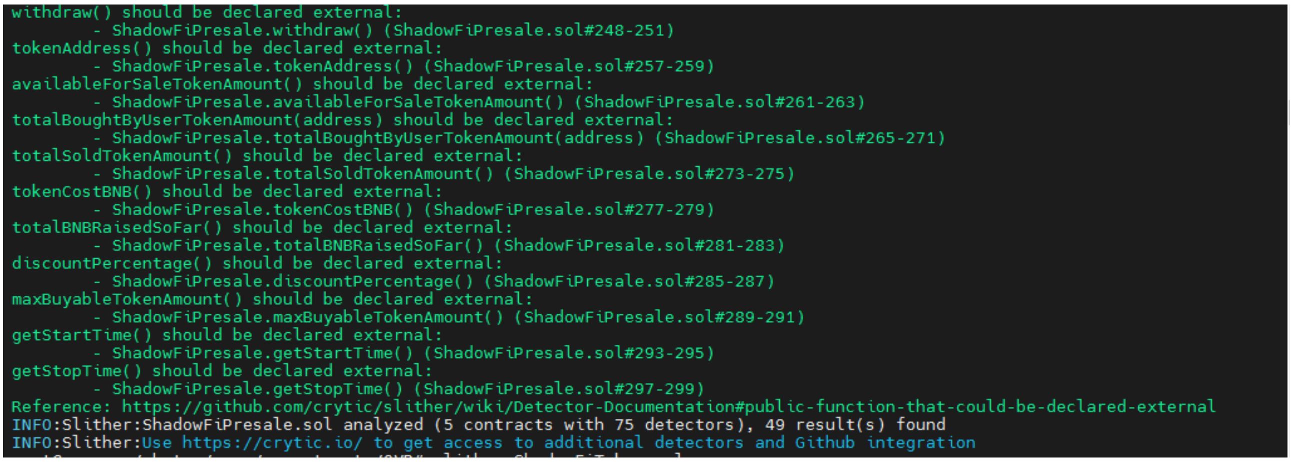
```
INFO:Detectors:
Reentrancy in ShadowFiPresale.buy(uint256) (ShadowFiPresale.sol#301-342):
        External calls:
        - token.transfer(address(msg.sender),_amount) (ShadowFiPresale.sol#334)
        External calls sending eth:
        - address(owner()).transfer(totalCost) (ShadowFiPresale.sol#327)
        - address(msg.sender).transfer(excess) (ShadowFiPresale.sol#331)
        State variables written after the call(s):
        - totalBNBRaised += totalCost (ShadowFiPresale.sol#339)
        - totalSold += _amount (ShadowFiPresale.sol#338)
Reentrancy in ShadowFiPresale.depositTokens(address,uint256) (ShadowFiPresale.sol#183-195):
        External calls:
        - token.transferFrom(address(msg.sender),address(this),_amount) (ShadowFiPresale.sol#193)
        State variables written after the call(s):
        - availableForSale += _amount (ShadowFiPresale.sol#194)
Reentrancy in ShadowFiPresale.withdrawTokens(address,uint256) (ShadowFiPresale.sol#197-209):
        External calls:
        - token.transfer(address(msg.sender),_amount) (ShadowFiPresale.sol#207)
        State variables written after the call(s):
        - availableForSale -= _amount (ShadowFiPresale.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in ShadowFiPresale.buy(uint256) (ShadowFiPresale.sol#301-342):
        External calls:
        - token.transfer(address(msg.sender),_amount) (ShadowFiPresale.sol#334)
        External calls sending eth:
        - address(owner()).transfer(totalCost) (ShadowFiPresale.sol#327)
        - address(msg.sender).transfer(excess) (ShadowFiPresale.sol#331)
        Event emitted after the call(s):
        - buyTokens(msg.sender,_amount,msg.value,discounted) (ShadowFiPresale.sol#341)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```



```
INFO:Detectors:
ShadowFiPresale.setStartandStopTime(uint32,uint32) (ShadowFiPresale.sol#230-242) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_stopTime > block.timestamp,Stop time must be before current time.) (ShadowFiPresale.sol#235-23
8)
ShadowFiPresale.buy(uint256) (ShadowFiPresale.sol#301-342) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= startTime,Presale has not started yet.) (ShadowFiPresale.sol#302)
        - require(bool,string)(block.timestamp <= stopTime,Presale has ended.) (ShadowFiPresale.sol#303)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Context._msgData() (ShadowFiPresale.sol#57-59) is never used and should be removed
ReentrancyGuard._nonReentrantAfter() (ShadowFiPresale.sol#45-49) is never used and should be removed
ReentrancyGuard._nonReentrantBefore() (ShadowFiPresale.sol#37-43) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

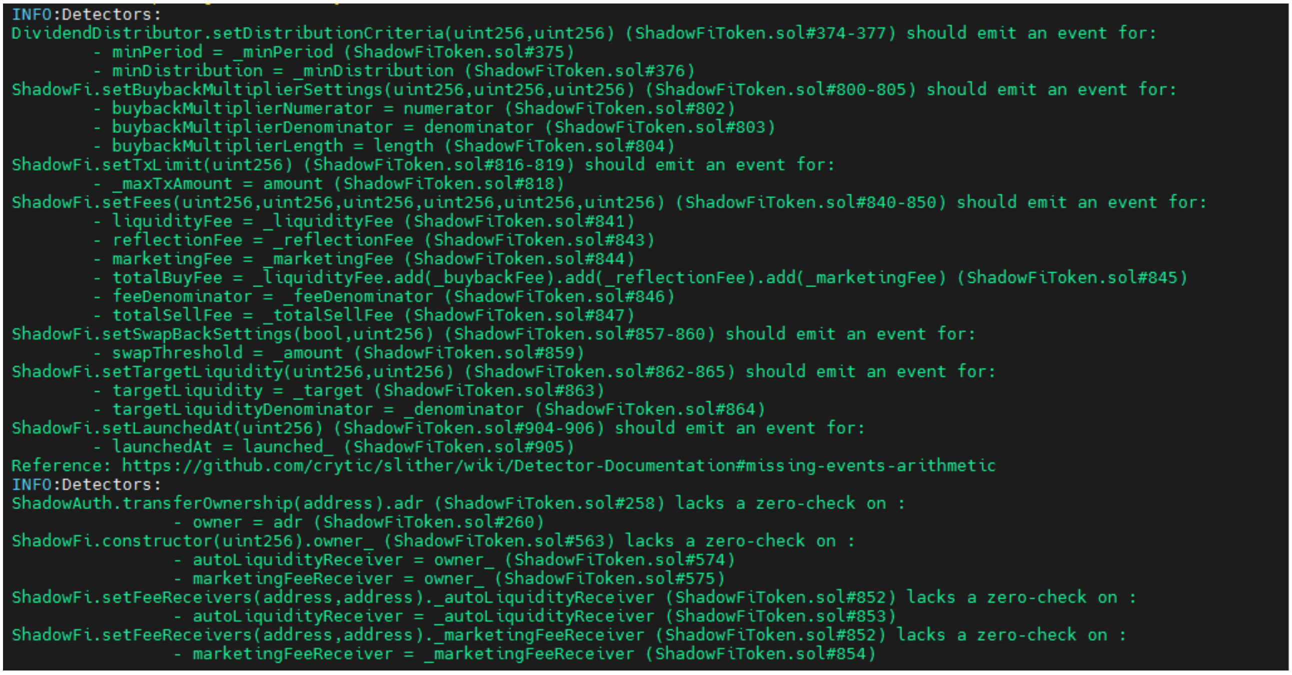


```
INFO:Detectors:
Reentrancy in ShadowFiPresale.buy(uint256) (ShadowFiPresale.sol#301-342):
        External calls:
        - address(owner()).transfer(totalCost) (ShadowFiPresale.sol#327)
        - address(msg.sender).transfer(excess) (ShadowFiPresale.sol#331)
        State variables written after the call(s):
        - availableForSale -= _amount (ShadowFiPresale.sol#337)
        - totalBNBRaised += totalCost (ShadowFiPresale.sol#339)
        - totalBoughtByUser[msg.sender] += _amount (ShadowFiPresale.sol#336)
        - totalSold += _amount (ShadowFiPresale.sol#338)
        Event emitted after the call(s):
        - buyTokens(msg.sender,_amount,msg.value,discounted) (ShadowFiPresale.sol#341)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (ShadowFiPresale.sol#96-102)
depositTokens(address,uint256) should be declared external:
        - ShadowFiPresale.depositTokens(address,uint256) (ShadowFiPresale.sol#183-195)
withdrawTokens(address,uint256) should be declared external:
        - ShadowFiPresale.withdrawTokens(address,uint256) (ShadowFiPresale.sol#197-209)
setCost(uint256) should be declared external:
        - ShadowFiPresale.setCost(uint256) (ShadowFiPresale.sol#211-213)
setToken(address) should be declared external:
        - ShadowFiPresale.setToken(address) (ShadowFiPresale.sol#215-219)
setDiscount(uint256) should be declared external:
        - ShadowFiPresale.setDiscount(uint256) (ShadowFiPresale.sol#221-228)
setStartandStopTime(uint32,uint32) should be declared external:
        - ShadowFiPresale.setStartandStopTime(uint32,uint32) (ShadowFiPresale.sol#230-242)
setMax(uint256) should be declared external:
        - ShadowFiPresale.setMax(uint256) (ShadowFiPresale.sol#244-246)
```

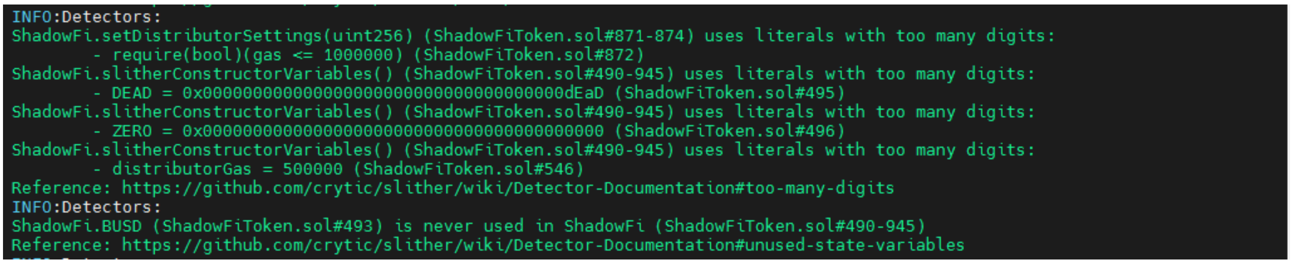


```
withdraw() should be declared external:
        - ShadowFiPresale.withdraw() (ShadowFiPresale.sol#248-251)
tokenAddress() should be declared external:
        - ShadowFiPresale.tokenAddress() (ShadowFiPresale.sol#257-259)
availableForSaleTokenAmount() should be declared external:
        - ShadowFiPresale.availableForSaleTokenAmount() (ShadowFiPresale.sol#261-263)
totalBoughtByUserTokenAmount(address) should be declared external:
        - ShadowFiPresale.totalBoughtByUserTokenAmount(address) (ShadowFiPresale.sol#265-271)
totalSoldTokenAmount() should be declared external:
        - ShadowFiPresale.totalSoldTokenAmount() (ShadowFiPresale.sol#273-275)
tokenCostBNB() should be declared external:
        - ShadowFiPresale.tokenCostBNB() (ShadowFiPresale.sol#277-279)
totalBNBRaisedSoFar() should be declared external:
        - ShadowFiPresale.totalBNBRaisedSoFar() (ShadowFiPresale.sol#281-283)
discountPercentage() should be declared external:
        - ShadowFiPresale.discountPercentage() (ShadowFiPresale.sol#285-287)
maxBuyableTokenAmount() should be declared external:
        - ShadowFiPresale.maxBuyableTokenAmount() (ShadowFiPresale.sol#289-291)
getStartTime() should be declared external:
        - ShadowFiPresale.getStartTime() (ShadowFiPresale.sol#293-295)
getStopTime() should be declared external:
        - ShadowFiPresale.getStopTime() (ShadowFiPresale.sol#297-299)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ShadowFiPresale.sol analyzed (5 contracts with 75 detectors), 49 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

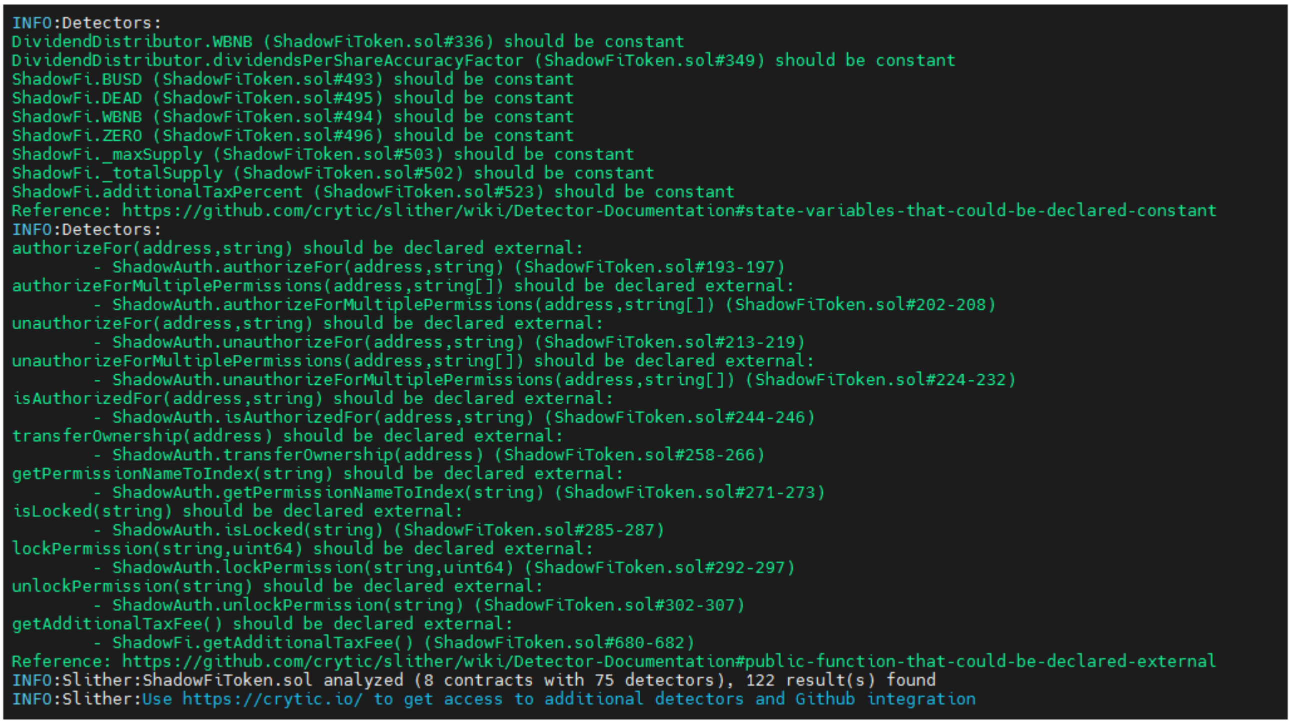
Slither log >> ShadowFiToken.sol



```
INFO:Detectors:
DividendDistributor.setDistributionCriteria(uint256,uint256) (ShadowFiToken.sol#374-377) should emit an event for:
        - minPeriod = _minPeriod (ShadowFiToken.sol#375)
        - minDistribution = _minDistribution (ShadowFiToken.sol#376)
ShadowFi.setBuybackMultiplierSettings(uint256,uint256,uint256) (ShadowFiToken.sol#800-805) should emit an event for:
        - buybackMultiplierNumerator = numerator (ShadowFiToken.sol#802)
        - buybackMultiplierDenominator = denominator (ShadowFiToken.sol#803)
        - buybackMultiplierLength = length (ShadowFiToken.sol#804)
ShadowFi.setTxLimit(uint256) (ShadowFiToken.sol#816-819) should emit an event for:
        - _maxTxAmount = amount (ShadowFiToken.sol#818)
ShadowFi.setFees(uint256,uint256,uint256,uint256,uint256) (ShadowFiToken.sol#840-850) should emit an event for:
        - liquidityFee = _liquidityFee (ShadowFiToken.sol#841)
        - reflectionFee = _reflectionFee (ShadowFiToken.sol#843)
        - marketingFee = _marketingFee (ShadowFiToken.sol#844)
        - totalBuyFee = _liquidityFee.add(_buybackFee).add(_reflectionFee).add(_marketingFee) (ShadowFiToken.sol#845)
        - feeDenominator = _feeDenominator (ShadowFiToken.sol#846)
        - totalSellFee = _totalSellFee (ShadowFiToken.sol#847)
ShadowFi.setSwapBackSettings(bool,uint256) (ShadowFiToken.sol#857-860) should emit an event for:
        - swapThreshold = _amount (ShadowFiToken.sol#859)
ShadowFi.setTargetLiquidity(uint256,uint256) (ShadowFiToken.sol#862-865) should emit an event for:
        - targetLiquidity = _target (ShadowFiToken.sol#863)
        - targetLiquidityDenominator = _denominator (ShadowFiToken.sol#864)
ShadowFi.setLaunchedAt(uint256) (ShadowFiToken.sol#904-906) should emit an event for:
        - launchedAt = launched_ (ShadowFiToken.sol#905)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
ShadowAuth.transferOwnership(address).adr (ShadowFiToken.sol#258) lacks a zero-check on :
                - owner = adr (ShadowFiToken.sol#260)
ShadowFi.constructor(uint256).owner_ (ShadowFiToken.sol#563) lacks a zero-check on :
                - autoLiquidityReceiver = owner_ (ShadowFiToken.sol#574)
                - marketingFeeReceiver = owner_ (ShadowFiToken.sol#575)
ShadowFi.setFeeReceivers(address,address)._autoLiquidityReceiver (ShadowFiToken.sol#852) lacks a zero-check on :
                - autoLiquidityReceiver = _autoLiquidityReceiver (ShadowFiToken.sol#853)
ShadowFi.setFeeReceivers(address,address)._marketingFeeReceiver (ShadowFiToken.sol#852) lacks a zero-check on :
                - marketingFeeReceiver = _marketingFeeReceiver (ShadowFiToken.sol#854)
```



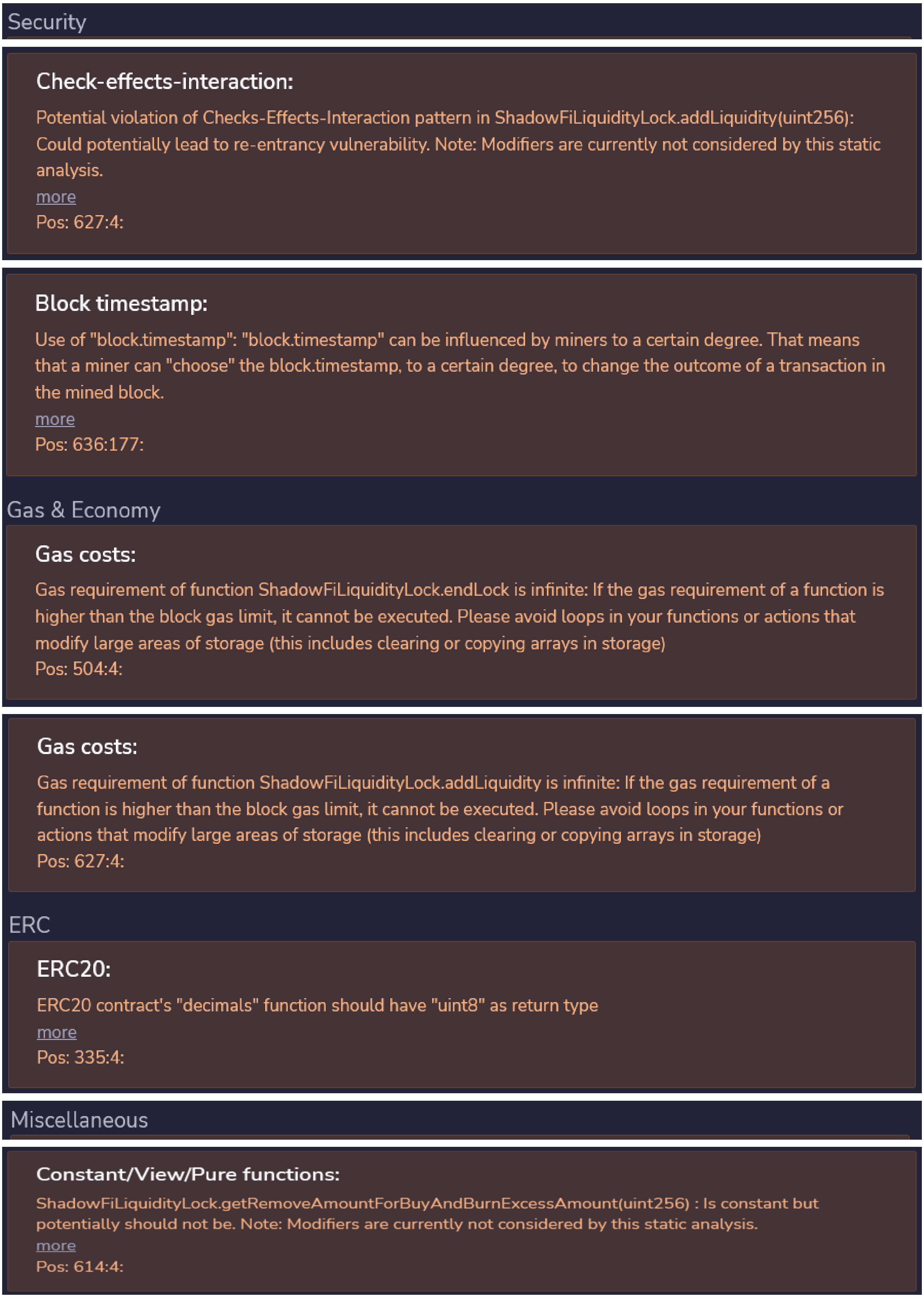
```
INFO:Detectors:
ShadowFi.setDistributorSettings(uint256) (ShadowFiToken.sol#871-874) uses literals with too many digits:
        - require(bool)(gas <= 1000000) (ShadowFiToken.sol#872)
ShadowFi.slitherConstructorVariables() (ShadowFiToken.sol#490-945) uses literals with too many digits:
        - DEAD = 0x000000000000000000000000000000000000dEaD (ShadowFiToken.sol#495)
ShadowFi.slitherConstructorVariables() (ShadowFiToken.sol#490-945) uses literals with too many digits:
        - ZERO = 0x0000000000000000000000000000000000000000 (ShadowFiToken.sol#496)
ShadowFi.slitherConstructorVariables() (ShadowFiToken.sol#490-945) uses literals with too many digits:
        - distributorGas = 500000 (ShadowFiToken.sol#546)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
ShadowFi.BUSD (ShadowFiToken.sol#493) is never used in ShadowFi (ShadowFiToken.sol#490-945)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
```



```
INFO:Detectors:
DividendDistributor.WBNB (ShadowFiToken.sol#336) should be constant
DividendDistributor.dividendsPerShareAccuracyFactor (ShadowFiToken.sol#349) should be constant
ShadowFi.BUSD (ShadowFiToken.sol#493) should be constant
ShadowFi.DEAD (ShadowFiToken.sol#495) should be constant
ShadowFi.WBNB (ShadowFiToken.sol#494) should be constant
ShadowFi.ZERO (ShadowFiToken.sol#496) should be constant
ShadowFi._maxSupply (ShadowFiToken.sol#503) should be constant
ShadowFi._totalSupply (ShadowFiToken.sol#502) should be constant
ShadowFi.additionalTaxPercent (ShadowFiToken.sol#523) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
authorizeFor(address,string) should be declared external:
        - ShadowAuth.authorizeFor(address,string) (ShadowFiToken.sol#193-197)
authorizeForMultiplePermissions(address,string[]) should be declared external:
        - ShadowAuth.authorizeForMultiplePermissions(address,string[]) (ShadowFiToken.sol#202-208)
unauthorizeFor(address,string) should be declared external:
        - ShadowAuth.unauthorizeFor(address,string) (ShadowFiToken.sol#213-219)
unauthorizeForMultiplePermissions(address,string[]) should be declared external:
        - ShadowAuth.unauthorizeForMultiplePermissions(address,string[]) (ShadowFiToken.sol#224-232)
isAuthorizedFor(address,string) should be declared external:
        - ShadowAuth.isAuthorizedFor(address,string) (ShadowFiToken.sol#244-246)
transferOwnership(address) should be declared external:
        - ShadowAuth.transferOwnership(address) (ShadowFiToken.sol#258-266)
getPermissionNameToIndex(string) should be declared external:
        - ShadowAuth.getPermissionNameToIndex(string) (ShadowFiToken.sol#271-273)
isLocked(string) should be declared external:
        - ShadowAuth.isLocked(string) (ShadowFiToken.sol#285-287)
lockPermission(string,uint64) should be declared external:
        - ShadowAuth.lockPermission(string,uint64) (ShadowFiToken.sol#292-297)
unlockPermission(string) should be declared external:
        - ShadowAuth.unlockPermission(string) (ShadowFiToken.sol#302-307)
getAdditionalTaxFee() should be declared external:
        - ShadowFi.getAdditionalTaxFee() (ShadowFiToken.sol#680-682)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ShadowFiToken.sol analyzed (8 contracts with 75 detectors), 122 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

ShadowFiLPVault.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in ShadowFiLiquidityLock.addLiquidity(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 627:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 636:177:

Gas & Economy

Gas costs:

Gas requirement of function ShadowFiLiquidityLock.endLock is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 504:4:

Gas costs:

Gas requirement of function ShadowFiLiquidityLock.addLiquidity is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 627:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type

more

Pos: 335:4:

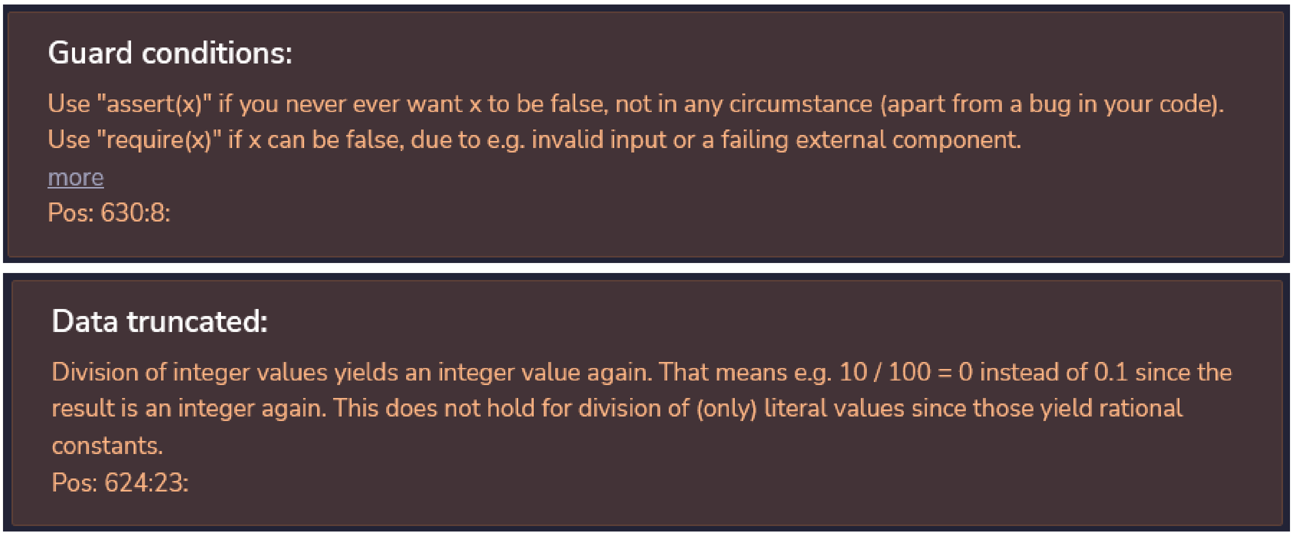
Miscellaneous

Constant/View/Pure functions:

ShadowFiLiquidityLock.getRemoveAmountForBuyAndBurnExcessAmount(uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 614:4:



Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

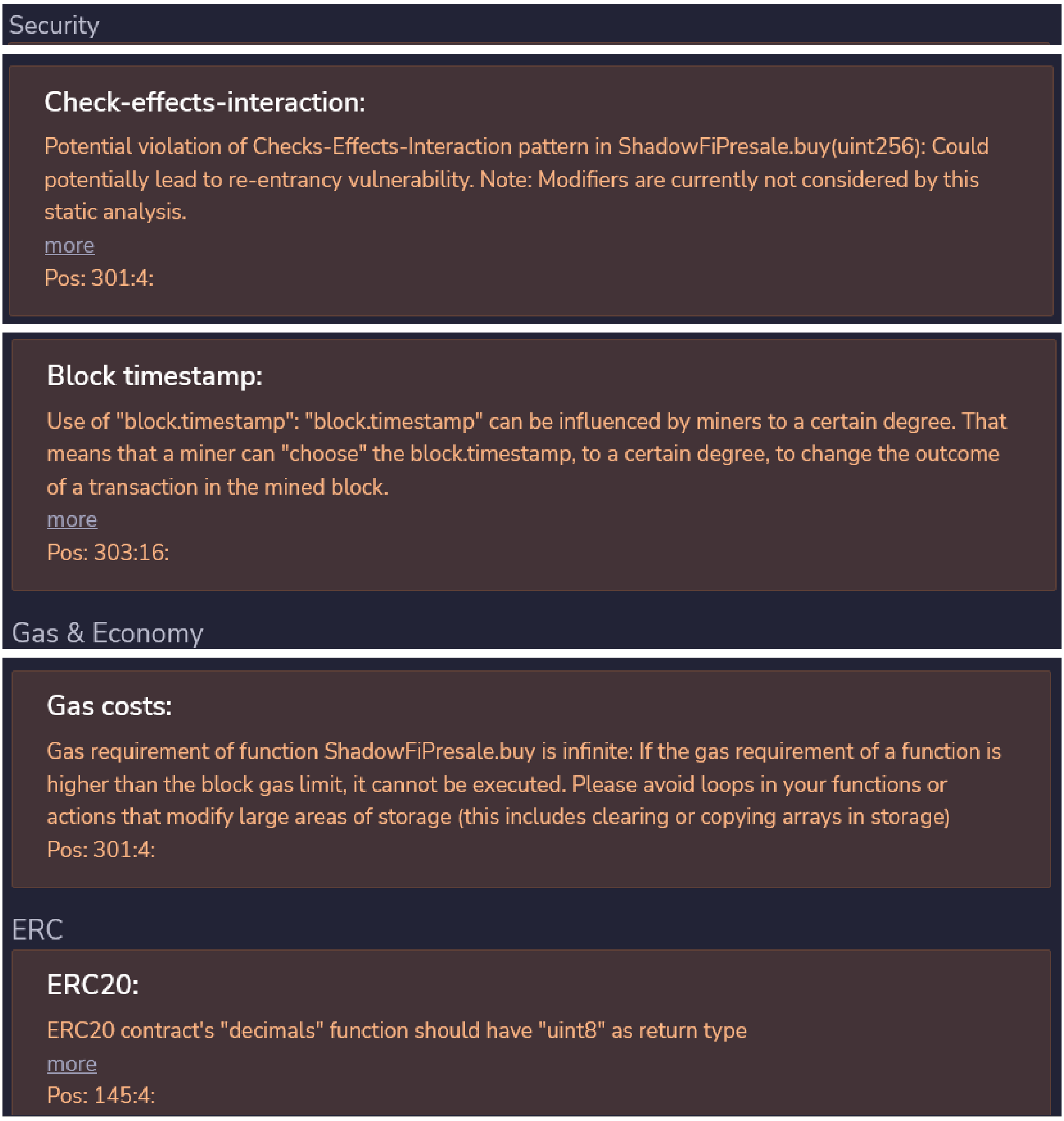
Pos: 630:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 624:23:

ShadowFiPresale.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in ShadowFiPresale.buy(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 301:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 303:16:

Gas & Economy

Gas costs:

Gas requirement of function ShadowFiPresale.buy is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 301:4:

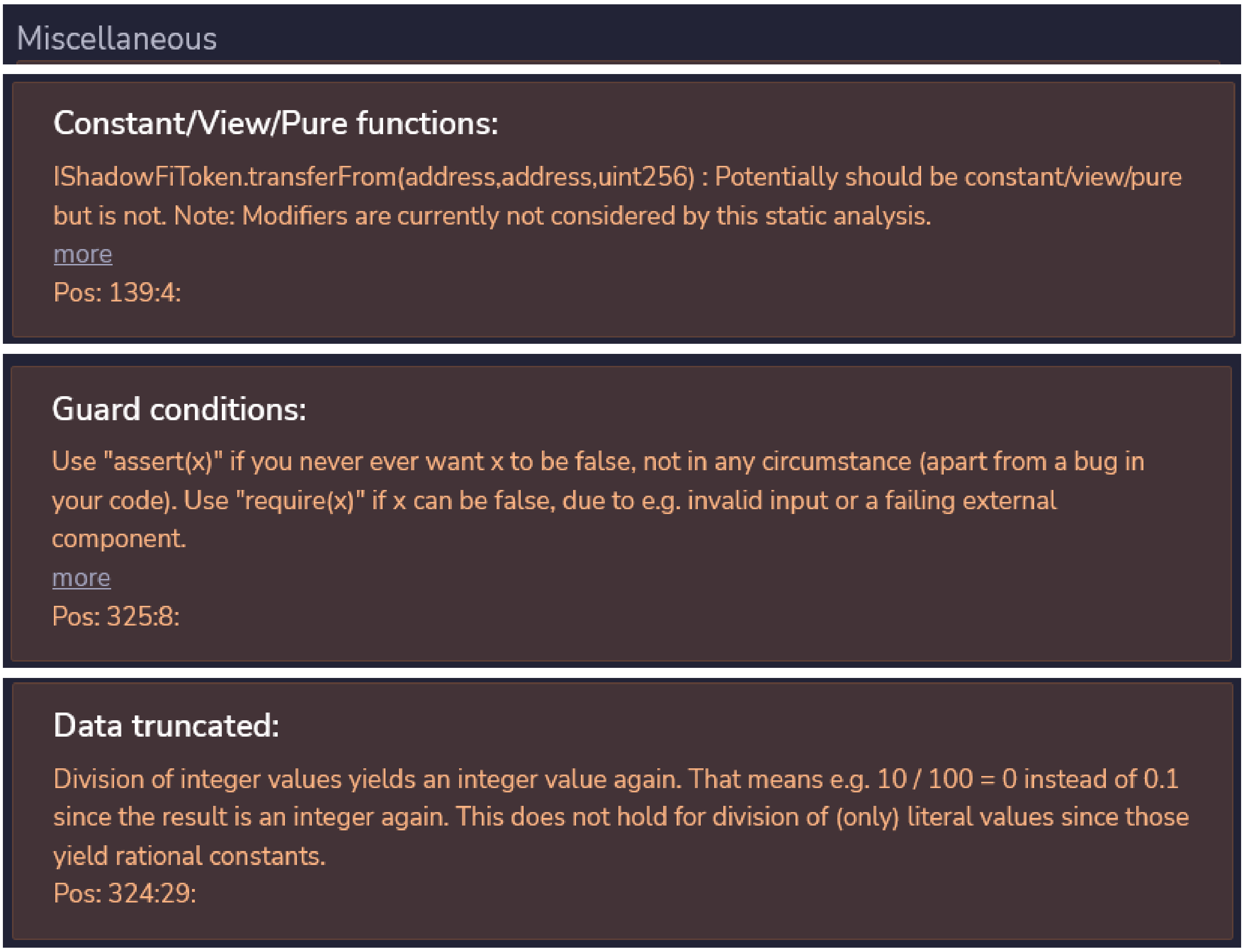
ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type

more

Pos: 145:4:



Miscellaneous

Constant/View/Pure functions:

IShadowFiToken.transferFrom(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 139:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

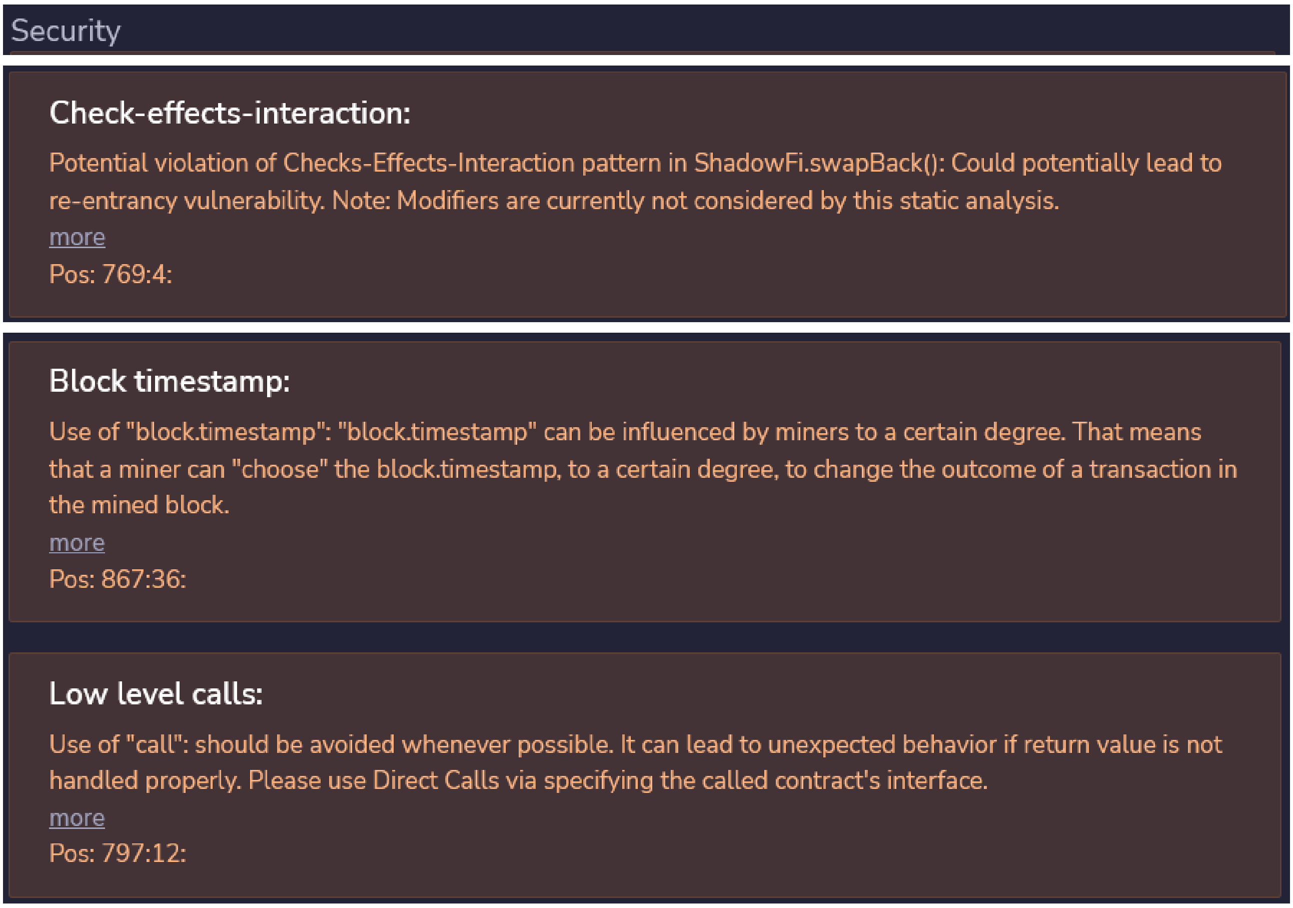
Pos: 325:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 324:29:

ShadowFiToken.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in ShadowFi.swapBack(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 769:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

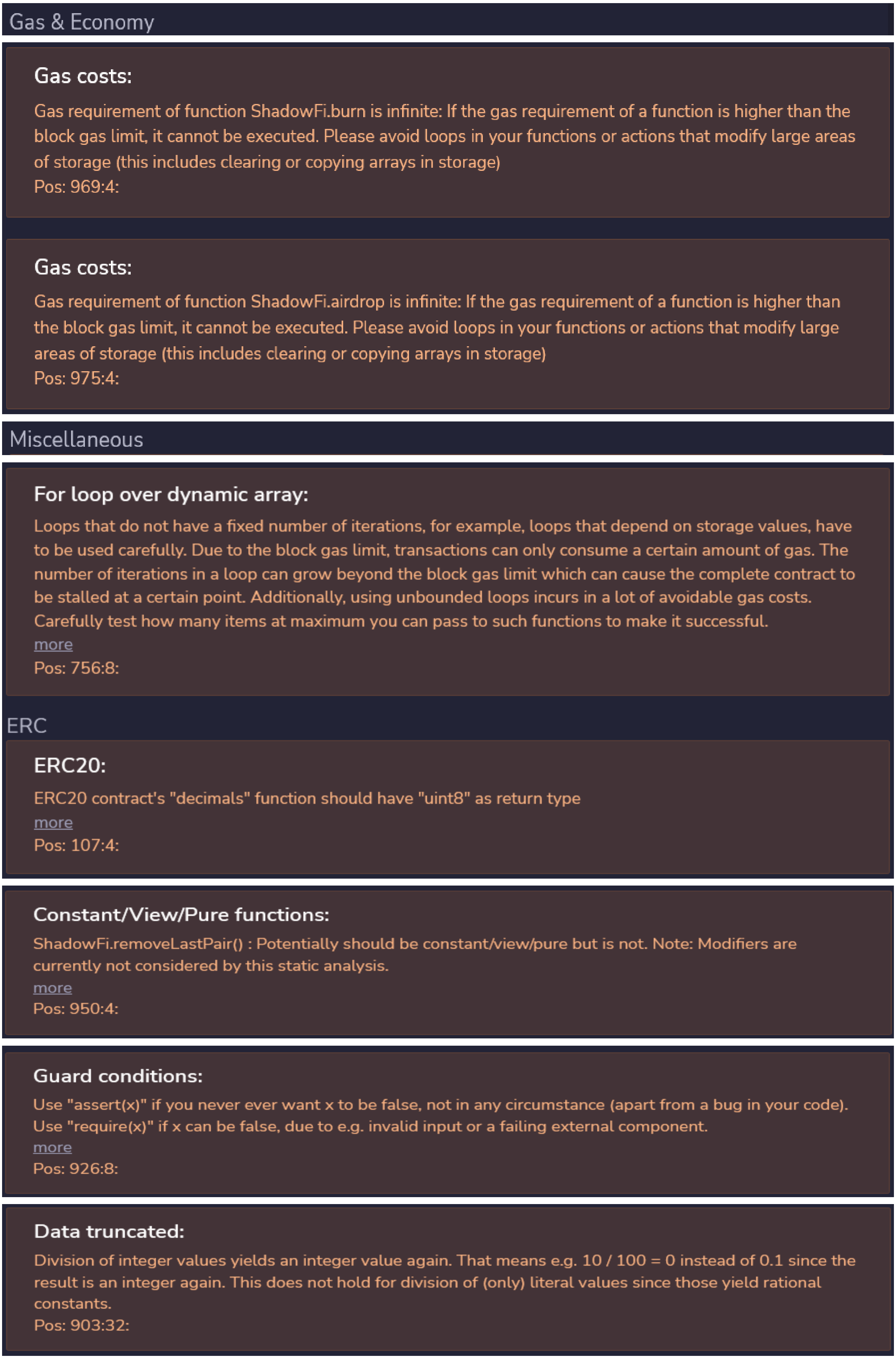
Pos: 867:36:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.

more

Pos: 797:12:



Gas & Economy

Gas costs:

Gas requirement of function ShadowFi.burn is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 969:4:

Gas costs:

Gas requirement of function ShadowFi.airdrop is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 975:4:

Miscellaneous

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 756:8:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 107:4:

Constant/View/Pure functions:

ShadowFi.removeLastPair() : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 950:4:

Guard conditions:

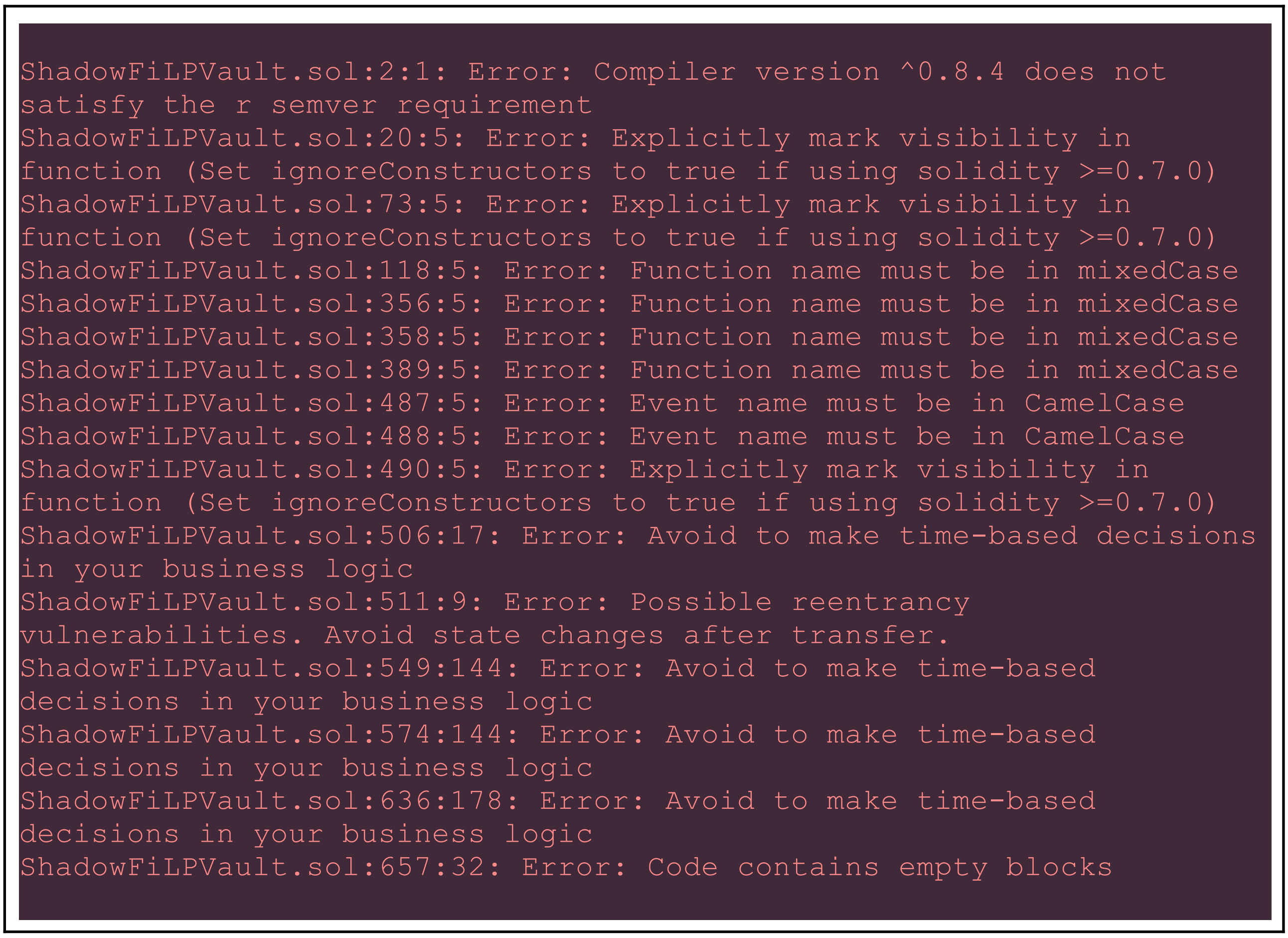
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 926:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 903:32:

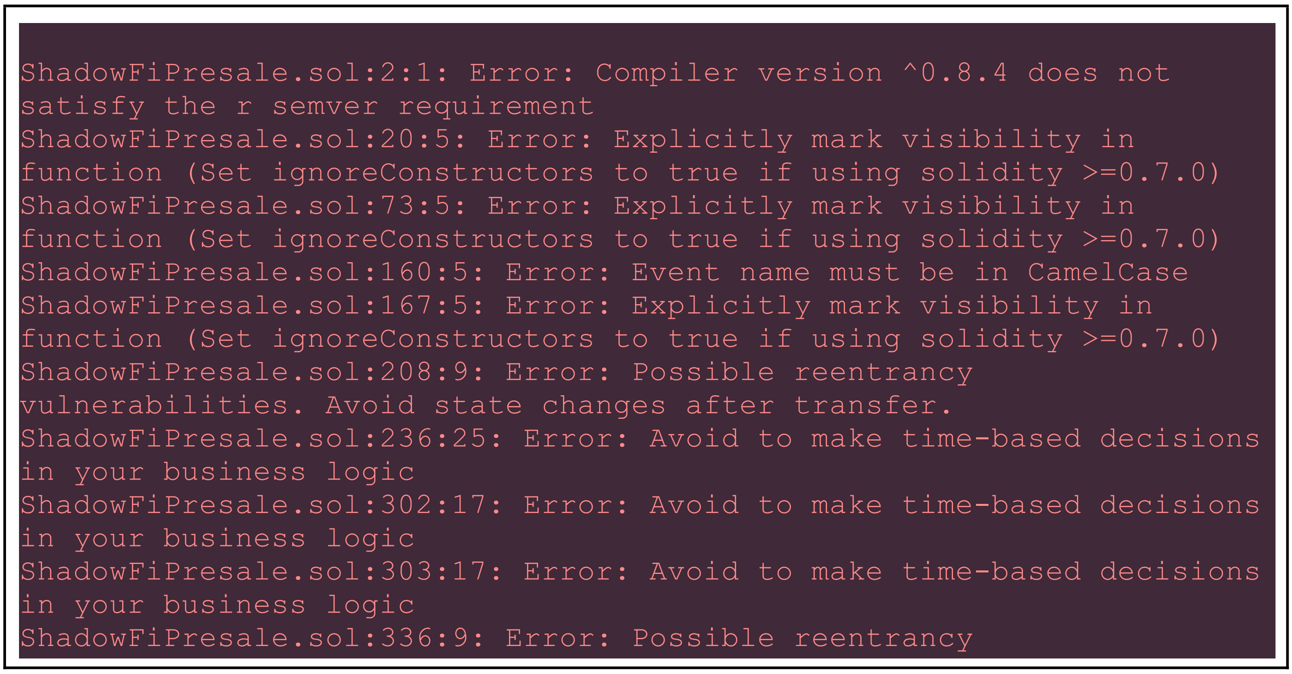
Solhint Linter

ShadowFiLPVault.sol

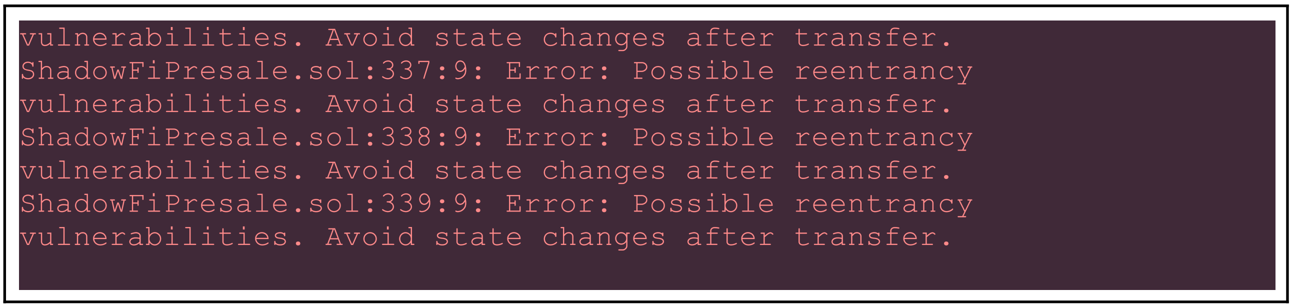


```
ShadowFiLPVault.sol:2:1: Error: Compiler version ^0.8.4 does not
satisfy the r semver requirement
ShadowFiLPVault.sol:20:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
ShadowFiLPVault.sol:73:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
ShadowFiLPVault.sol:118:5: Error: Function name must be in mixedCase
ShadowFiLPVault.sol:356:5: Error: Function name must be in mixedCase
ShadowFiLPVault.sol:358:5: Error: Function name must be in mixedCase
ShadowFiLPVault.sol:389:5: Error: Function name must be in mixedCase
ShadowFiLPVault.sol:487:5: Error: Event name must be in CamelCase
ShadowFiLPVault.sol:488:5: Error: Event name must be in CamelCase
ShadowFiLPVault.sol:490:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
ShadowFiLPVault.sol:506:17: Error: Avoid to make time-based decisions
in your business logic
ShadowFiLPVault.sol:511:9: Error: Possible reentrancy
vulnerabilities. Avoid state changes after transfer.
ShadowFiLPVault.sol:549:144: Error: Avoid to make time-based
decisions in your business logic
ShadowFiLPVault.sol:574:144: Error: Avoid to make time-based
decisions in your business logic
ShadowFiLPVault.sol:636:178: Error: Avoid to make time-based
decisions in your business logic
ShadowFiLPVault.sol:657:32: Error: Code contains empty blocks
```

ShadowFiPresale.sol

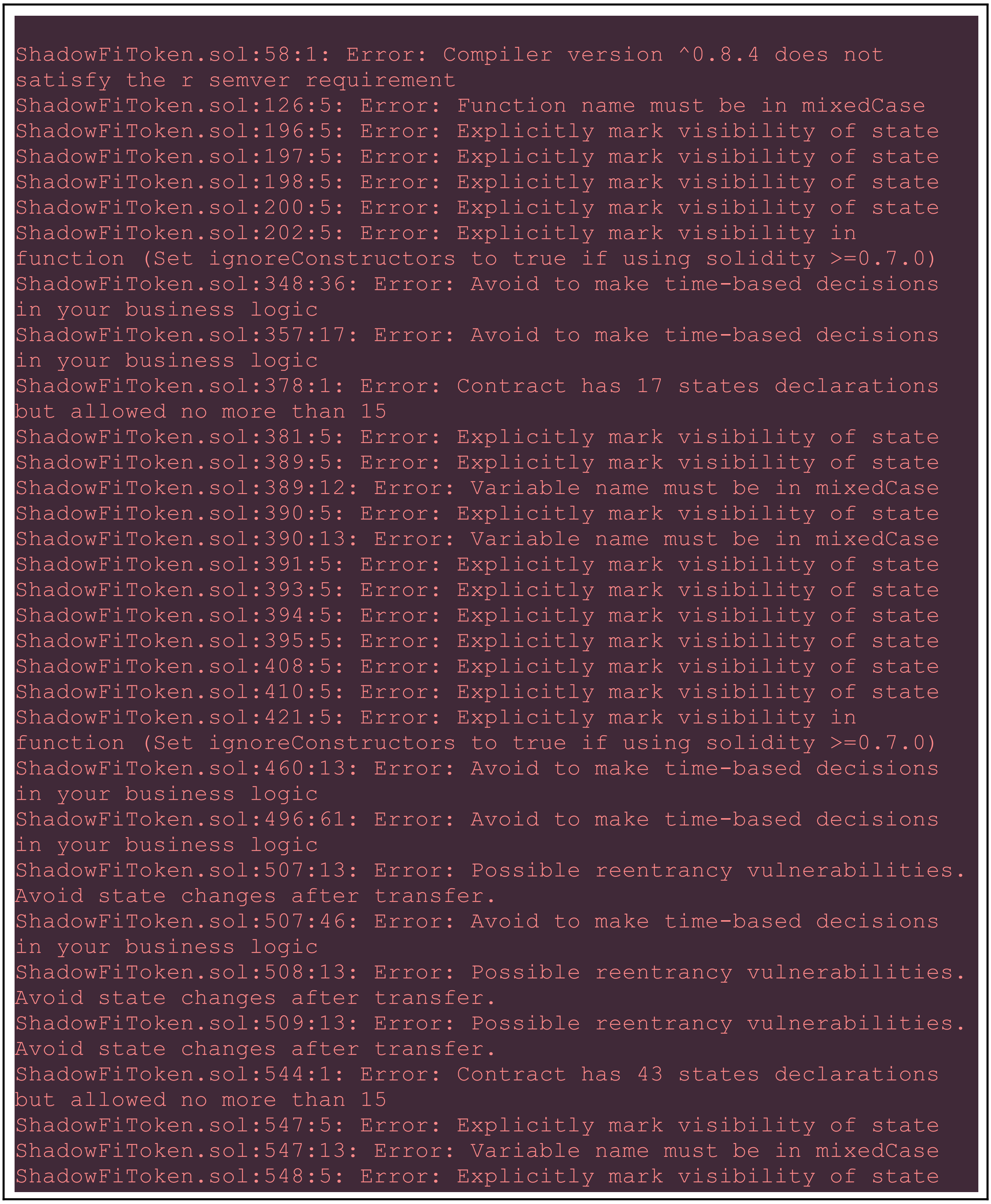


```
ShadowFiPresale.sol:2:1: Error: Compiler version ^0.8.4 does not
satisfy the r semver requirement
ShadowFiPresale.sol:20:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
ShadowFiPresale.sol:73:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
ShadowFiPresale.sol:160:5: Error: Event name must be in CamelCase
ShadowFiPresale.sol:167:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
ShadowFiPresale.sol:208:9: Error: Possible reentrancy
vulnerabilities. Avoid state changes after transfer.
ShadowFiPresale.sol:236:25: Error: Avoid to make time-based decisions
in your business logic
ShadowFiPresale.sol:302:17: Error: Avoid to make time-based decisions
in your business logic
ShadowFiPresale.sol:303:17: Error: Avoid to make time-based decisions
in your business logic
ShadowFiPresale.sol:336:9: Error: Possible reentrancy
```

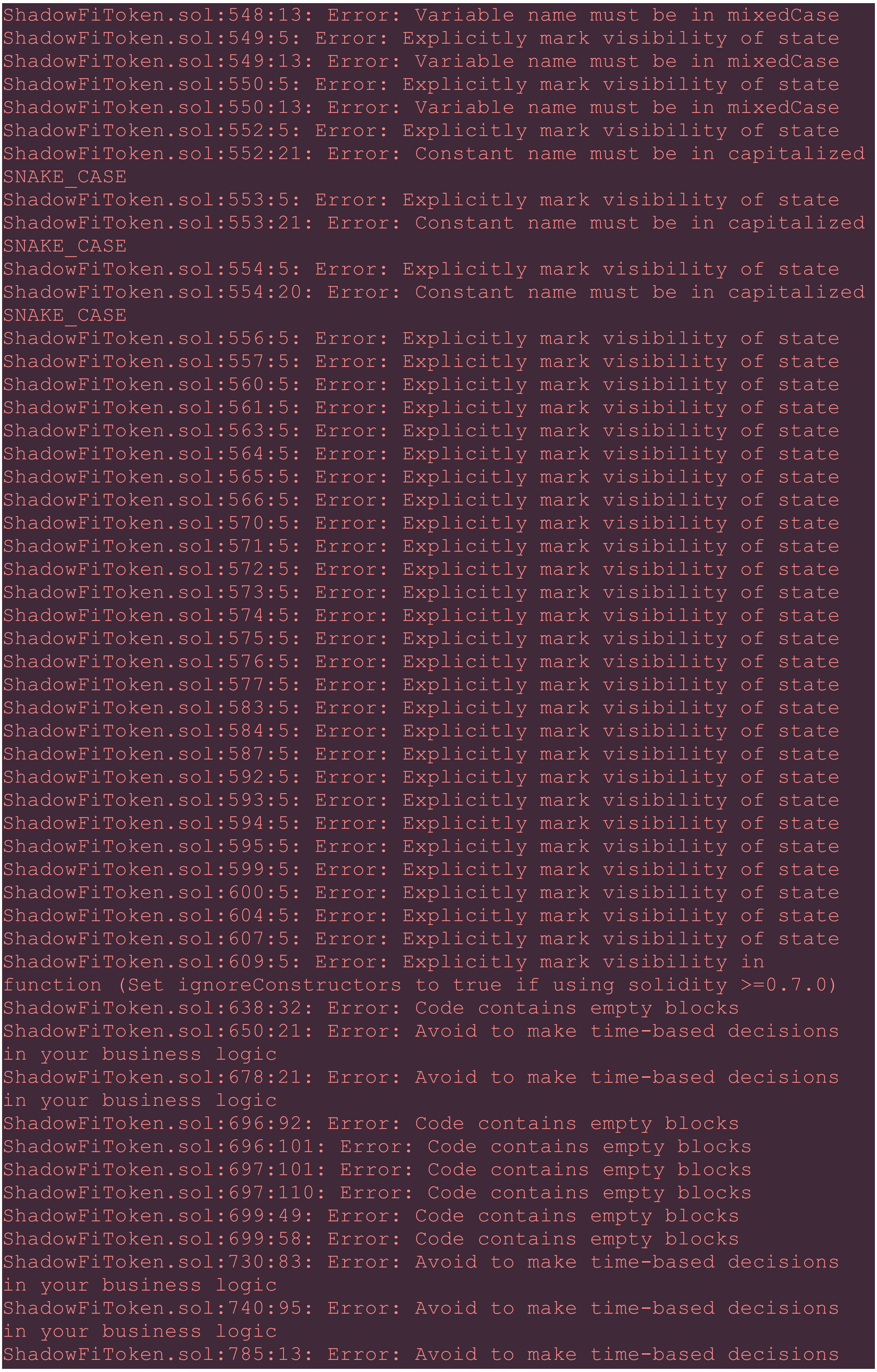


```
vulnerabilities. Avoid state changes after transfer.
ShadowFiPresale.sol:337:9: Error: Possible reentrancy
vulnerabilities. Avoid state changes after transfer.
ShadowFiPresale.sol:338:9: Error: Possible reentrancy
vulnerabilities. Avoid state changes after transfer.
ShadowFiPresale.sol:339:9: Error: Possible reentrancy
vulnerabilities. Avoid state changes after transfer.
```

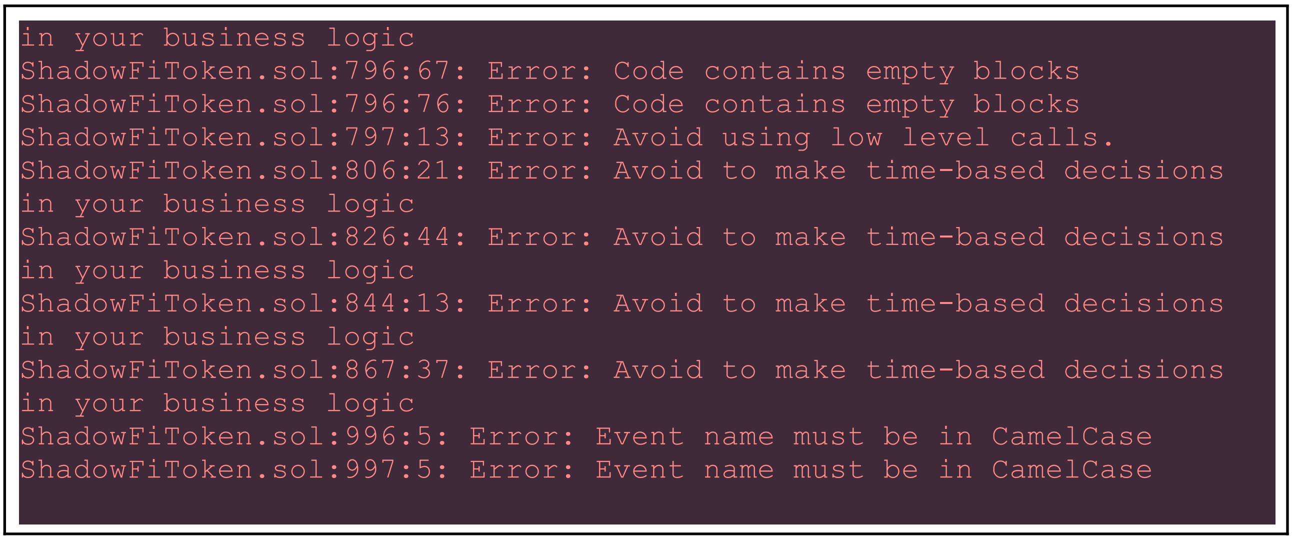
ShadowFiToken.sol



```
ShadowFiToken.sol:58:1: Error: Compiler version ^0.8.4 does not
satisfy the r semver requirement
ShadowFiToken.sol:126:5: Error: Function name must be in mixedCase
ShadowFiToken.sol:196:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:197:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:198:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:200:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:202:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
ShadowFiToken.sol:348:36: Error: Avoid to make time-based decisions
in your business logic
ShadowFiToken.sol:357:17: Error: Avoid to make time-based decisions
in your business logic
ShadowFiToken.sol:378:1: Error: Contract has 17 states declarations
but allowed no more than 15
ShadowFiToken.sol:381:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:389:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:389:12: Error: Variable name must be in mixedCase
ShadowFiToken.sol:390:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:390:13: Error: Variable name must be in mixedCase
ShadowFiToken.sol:391:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:393:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:394:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:395:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:408:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:410:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:421:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
ShadowFiToken.sol:460:13: Error: Avoid to make time-based decisions
in your business logic
ShadowFiToken.sol:496:61: Error: Avoid to make time-based decisions
in your business logic
ShadowFiToken.sol:507:13: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
ShadowFiToken.sol:507:46: Error: Avoid to make time-based decisions
in your business logic
ShadowFiToken.sol:508:13: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
ShadowFiToken.sol:509:13: Error: Possible reentrancy vulnerabilities.
Avoid state changes after transfer.
ShadowFiToken.sol:544:1: Error: Contract has 43 states declarations
but allowed no more than 15
ShadowFiToken.sol:547:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:547:13: Error: Variable name must be in mixedCase
ShadowFiToken.sol:548:5: Error: Explicitly mark visibility of state
```



```
ShadowFiToken.sol:548:13: Error: Variable name must be in mixedCase
ShadowFiToken.sol:549:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:549:13: Error: Variable name must be in mixedCase
ShadowFiToken.sol:550:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:550:13: Error: Variable name must be in mixedCase
ShadowFiToken.sol:552:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:552:21: Error: Constant name must be in capitalized
SNAKE_CASE
ShadowFiToken.sol:553:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:553:21: Error: Constant name must be in capitalized
SNAKE_CASE
ShadowFiToken.sol:554:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:554:20: Error: Constant name must be in capitalized
SNAKE_CASE
ShadowFiToken.sol:556:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:557:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:560:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:561:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:563:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:564:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:565:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:566:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:570:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:571:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:572:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:573:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:574:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:575:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:576:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:577:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:583:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:584:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:587:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:592:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:593:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:594:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:595:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:599:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:600:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:604:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:607:5: Error: Explicitly mark visibility of state
ShadowFiToken.sol:609:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
ShadowFiToken.sol:638:32: Error: Code contains empty blocks
ShadowFiToken.sol:650:21: Error: Avoid to make time-based decisions
in your business logic
ShadowFiToken.sol:678:21: Error: Avoid to make time-based decisions
in your business logic
ShadowFiToken.sol:696:92: Error: Code contains empty blocks
ShadowFiToken.sol:696:101: Error: Code contains empty blocks
ShadowFiToken.sol:697:101: Error: Code contains empty blocks
ShadowFiToken.sol:697:110: Error: Code contains empty blocks
ShadowFiToken.sol:699:49: Error: Code contains empty blocks
ShadowFiToken.sol:699:58: Error: Code contains empty blocks
ShadowFiToken.sol:730:83: Error: Avoid to make time-based decisions
in your business logic
ShadowFiToken.sol:740:95: Error: Avoid to make time-based decisions
in your business logic
ShadowFiToken.sol:785:13: Error: Avoid to make time-based decisions
```



```
in your business logic
ShadowFiToken.sol:796:67: Error: Code contains empty blocks
ShadowFiToken.sol:796:76: Error: Code contains empty blocks
ShadowFiToken.sol:797:13: Error: Avoid using low level calls.
ShadowFiToken.sol:806:21: Error: Avoid to make time-based decisions
in your business logic
ShadowFiToken.sol:826:44: Error: Avoid to make time-based decisions
in your business logic
ShadowFiToken.sol:844:13: Error: Avoid to make time-based decisions
in your business logic
ShadowFiToken.sol:867:37: Error: Avoid to make time-based decisions
in your business logic
ShadowFiToken.sol:996:5: Error: Event name must be in CamelCase
ShadowFiToken.sol:997:5: Error: Event name must be in CamelCase
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

EtherAuthority

Email: audit@EtherAuthority.io