EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Frax Token
Website:	frax.finance
Platform:	Ethereum
Language:	Solidity
Date:	April 29th, 2024

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

As part of EtherAuthority's community smart contracts audit initiatives, the smart contracts of Frax Token from frax.finance were audited. 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 April 29th, 2024.

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

- This contract leverages a combination of decentralized oracles, governance control, and access control mechanisms to maintain and adjust the collateralization and supply of the FRAX stablecoin dynamically. The design ensures that only authorized entities can mint and burn FRAX, and that the system parameters can be adjusted in a controlled manner to respond to market conditions.
- The provided Solidity code defines a smart contract named `FRAXStablecoin`, which extends the `ERC20Custom` and `AccessControl` contracts. This contract is designed for the FRAX stablecoin system, incorporating features such as dynamic collateral ratio adjustment, minting, and burning of tokens, and integration with Chainlink and Uniswap oracles for price feeds.
- The token is without any other custom functionality and without any ownership control, which makes it truly decentralized.
- Overall, this contract implements a stablecoin with dynamic collateralization ratio adjustments based on the price of FRAX. It also provides functionalities for interacting with pools and managing parameters like fees, oracles, and permissions.

Audit scope

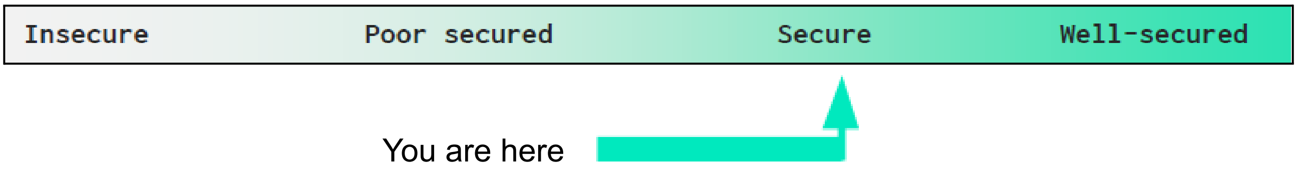
Name	**Code Review and Security Analysis Report for Frax Token Smart Contract**
Platform	**Ethereum**
File	FRAXStablecoin.sol
Smart Contract Code	0x853d955acef822db058eb8505911ed77f175b99e
Audit Date	April 29th, 2024

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: • Name: Frax • Symbol: FRAX • Decimals: 18	**YES, This is valid.**
Owner/Governance control: • Add/remove the pool address. • Update a new owner's address. • Update redemption fee and minting fee. • Update frax step. • Update price target. • Update Cooldown value. • Update FXS address. • Sets the FXS_ETH Uniswap Oracle address. • Sets the FRAX_ETH Uniswap Oracle address.	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, the Customer`s solidity-based smart contracts are **“Secured”.** Also, these contracts 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 a 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. A general overview is presented in the AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium, 1 low, and 3 very low-level issues.

Investor Advice: A 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	The solidity version is not specified	Passed
	The solidity version is too old	Moderated
	Integer overflow/underflow	Passed
	Function input parameters lack 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	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	Passed
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 is not set	Passed
	"Short Address" Attack	Passed
	"Double Spend" Attack	Passed

Overall Audit Result: PASSED

Business Risk Analysis

Category	Result
🟢 Buy Tax	0%
🟢 Sell Tax	0%
🟢 Cannot Buy	No
🟢 Cannot Sell	No
🟢 Max Tax	0%
🟢 Modify Tax	Not Detected
🟢 Fee Check	No
🟢 Is Honeypot	Not Detected
🟢 Trading Cooldown	Not Detected
🟢 Can Pause Trade?	No
🟢 Pause Transfer?	No
🟢 Max Tax?	No
🟢 Is it Anti-whale?	No
🟢 Is Anti-bot?	Not Detected
🟢 Is it a Blacklist?	Not Detected
🟢 Blacklist Check	No
🟢 Can Mint?	Yes
🟢 Is it a Proxy?	No
🟢 Can Take Ownership?	No
🟢 Hidden Owner?	No
🟢 Self Destruction?	No
🟢 Auditor Confidence	High

Overall Audit Result: PASSED

Code Quality

This audit scope has 1 smart contract. Smart contracts contain Libraries, Smart contracts, inherits, and Interfaces. This is a compact and well-written smart contract.

The libraries in Frax Token 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 Frax Token.

The EtherAuthority team has no scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are well commented on in the smart contracts. Ethereum's NatSpec commenting style is recommended.

Documentation

We were given a Frax Token smart contract code in the form of an Etherscan web link.

As mentioned above, code parts are well commented on. 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.

Use of Dependencies

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

Apart from libraries, its functions are not used in external smart contract calls.

AS-IS overview

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onlyCollateralRatioPauser	modifier	Passed	No Issue
3	onlyPools	modifier	Passed	No Issue
4	onlyByOwnerOrGovernance	modifier	Passed	No Issue
5	onlyByOwnerGovernanceOrPool	modifier	Passed	No Issue
6	oracle_price	internal	Passed	No Issue
7	frax_price	read	Passed	No Issue
8	fxs_price	read	Passed	No Issue
9	eth_usd_price	read	Passed	No Issue
10	frax_info	read	Passed	No Issue
11	globalCollateralValue	read	Passed	No Issue
12	refreshCollateralRatio	write	Passed	No Issue
13	pool_burn_from	write	access only Pools	No Issue
14	pool_mint	write	access only Pools	No Issue
15	addPool	write	access only By Owner Or Governance	No Issue
16	removePool	write	access only By Owner Or Governance	No Issue
17	setOwner	write	Critical operation lacks event log, Missing Zero Address Validation	Refer Audit Findings
18	setRedemptionFee	write	Critical operation lacks event log	Refer Audit Findings
19	setMintingFee	write	Critical operation lacks event log	Refer Audit Findings
20	setFraxStep	write	Critical operation lacks event log	Refer Audit Findings
21	setPriceTarget	write	Critical operation lacks event log	Refer Audit Findings
22	setRefreshCooldown	write	access only By Owner Or Governance	No Issue
23	setFXSAddress	write	Critical operation lacks event log, Missing Zero Address Validation	Refer Audit Findings
24	setETHUSDOracle	write	Critical operation lacks event log	Refer Audit Findings

25	setTimelock	external	Critical operation lacks event log, Missing Zero Address Validation	Refer Audit Findings
26	setController	external	Critical operation lacks event log, Missing Zero Address Validation	Refer Audit Findings
27	setPriceBand	external	access only By Owner Or Governance	No Issue
28	setFRAXEthOracle	write	Missing Zero Address Validation, Missing Zero Address Validation	Refer Audit Findings
29	setFXSEthOracle	write	Missing Zero Address Validation	Refer Audit Findings
30	toggleCollateralRatio	write	access only Collateral Ratio Pauser	No Issue
41	totalSupply	read	Passed	No Issue
42	balanceOf	read	Passed	No Issue
43	transfer	write	Passed	No Issue
44	allowance	read	Passed	No Issue
45	approve	write	Passed	No Issue
46	transferFrom	write	Passed	No Issue
47	increaseAllowance	write	Passed	No Issue
48	decreaseAllowance	write	Passed	No Issue
49	_transfer	internal	Passed	No Issue
50	_mint	internal	Passed	No Issue
51	burn	write	Passed	No Issue
52	burnFrom	write	Passed	No Issue
53	_burn	internal	Passed	No Issue
54	_approve	internal	Passed	No Issue
55	_burnFrom	internal	Passed	No Issue
56	_beforeTokenTransfer	internal	Passed	No Issue
57	hasRole	read	Passed	No Issue
58	getRoleMemberCount	read	Passed	No Issue
59	getRoleMember	read	Passed	No Issue
60	getRoleAdmin	read	Passed	No Issue
61	grantRole	write	Passed	No Issue
62	revokeRole	write	Passed	No Issue
63	renounceRole	write	Passed	No Issue
64	_setupRole	internal	Passed	No Issue
65	_setRoleAdmin	internal	Passed	No Issue
66	_grantRole	write	Passed	No Issue
67	_revokeRole	write	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 a 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 lose
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets, that can't have a 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) Critical operation lacks event log:

Missing event log for :

- setOwner
- setRedemptionFee
- setMintingFee
- setFraxStep
- setPriceTarget
- setFXSAddress
- setETHUSDOracle
- setTimelock
- setPriceBand
- setFRAXEthOracle
- setFXSEthOracle
- setController.

Resolution: Write an event log for listed events.

Very Low / Informational / Best practices:

(1) Use the latest solidity version:

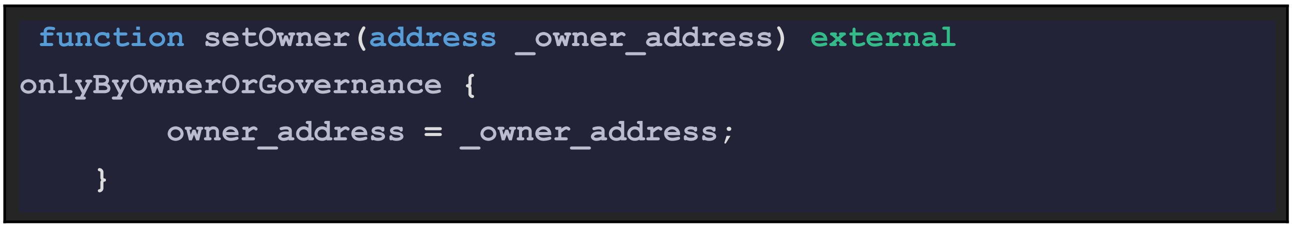
```
// SPDX-License-Identifier: MIT
pragma solidity 0.6.11;
```

Using the latest solidity will prevent any compiler-level bugs.

Resolution: Please use 0.8.25 which is the latest version.

(2) Missing Zero Address Validation:

```
// Sets the FXS_ETH Uniswap oracle address
    function setFXSEthOracle(address _fxs_oracle_addr, address
_weth_address) public onlyByOwnerOrGovernance {
        fxs_eth_oracle_address = _fxs_oracle_addr;
        fxsEthOracle = UniswapPairOracle(_fxs_oracle_addr);
        weth_address = _weth_address;
    }
 function setTimelock(address new_timelock) external
onlyByOwnerOrGovernance {
        timelock_address = new_timelock;
    }
function setController(address _controller_address) external
onlyByOwnerOrGovernance {
        controller_address = _controller_address;
    }
function setFXSAddress(address _fxs_address) public
onlyByOwnerOrGovernance {
        fxs_address = _fxs_address;
    }
function setETHUSDOracle(address _eth_usd_consumer_address) public
onlyByOwnerOrGovernance {
        eth_usd_consumer_address = _eth_usd_consumer_address;
        eth_usd_pricer =
ChainlinkETHUSDPriceConsumer(eth_usd_consumer_address);
        eth_usd_pricer_decimals = eth_usd_pricer.getDecimals();
    }
```



```
 function setOwner(address _owner_address) external
onlyByOwnerOrGovernance {
        owner_address = _owner_address;
    }
```

Addresses are not validated before assignment or external calls, potentially allowing the use of zero addresses and leading to unexpected behavior or vulnerabilities.

Resolution: It is recommended to add a zero-check for the passed-in address value to prevent unexpected errors.

(3) Variable mutability: **FRAXStablecoin.sol**

There are "name", "symbol", "creator_address" and "decimals" variables that are defined with immutability.

Resolution: We suggest defining the variable with the "private" keyword.

Centralization

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

FRAXStablecoin.sol

- pool_burn_from: The pool owner can redeem the amount.
- pool_mint: Pool owner can mint new FRAX.
- addPool: Adds collateral addresses by only the owner or governance.
- removePool: Remove a pool addressed by only the owner or governance.
- setOwner: The new owner address can be set by only the owner or governance.
- setRedemptionFee: The redemption fee can be set by only the owner or governance.
- setMintingFee: The minting fee can be set by only the owner or governance.
- setFraxStep: The new Frax Step value can be set by only the owner or governance.
- setPriceTarget: Price target value can be set by only the owner or governance.
- setRefreshCooldown: Refresh Cooldown value can be set by only the owner or governance.
- setFXSAddress: FXS addresses can be by only the owner or governance.
- setETHUSDOracle: Ether USD Oracle addresses can be by only the owner or governance.
- setTimelock: Timelock address can be set by only the owner or governance.
- setController: Controller address can be set by only the owner or governance.
- setPriceBand: Price Band value can be set by only the owner or governance.
- setFRAXEthOracle: The FRAX_ETH Uniswap oracle address can be set by only the owner or governance.
- setFXSEthOracle: The FXS_ETH Uniswap oracle address can be set by only the owner or governance.
- toggleCollateralRatio: Collateral Ratio can be toggled by the Collateral Ratio Pauser owner.

AccessControl.sol

- grantRole: Grants `role` to `account` by the admin role.
- revokeRole: Revokes `role` from `account` by the admin role.
- renounceRole: Revokes `role` from the calling account by the admin role.

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 Etherscan web links. And we have used all possible tests based on given objects as files. We observed 1 low and 3 informational issues in the smart contracts. And those issues are not critical. So, **it's 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 the 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.

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

Security state of the reviewed smart 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).

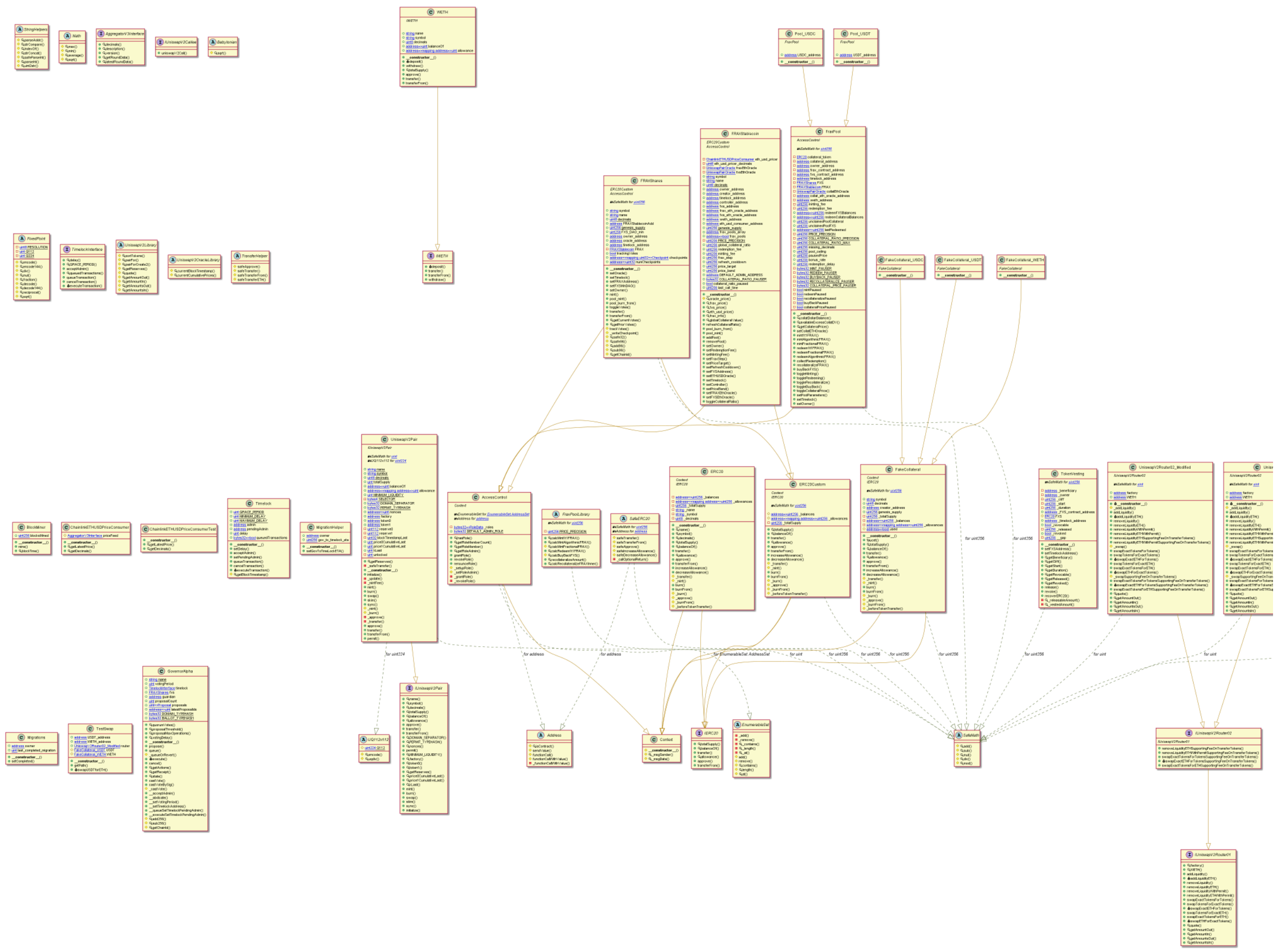
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we 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 - Frax Token

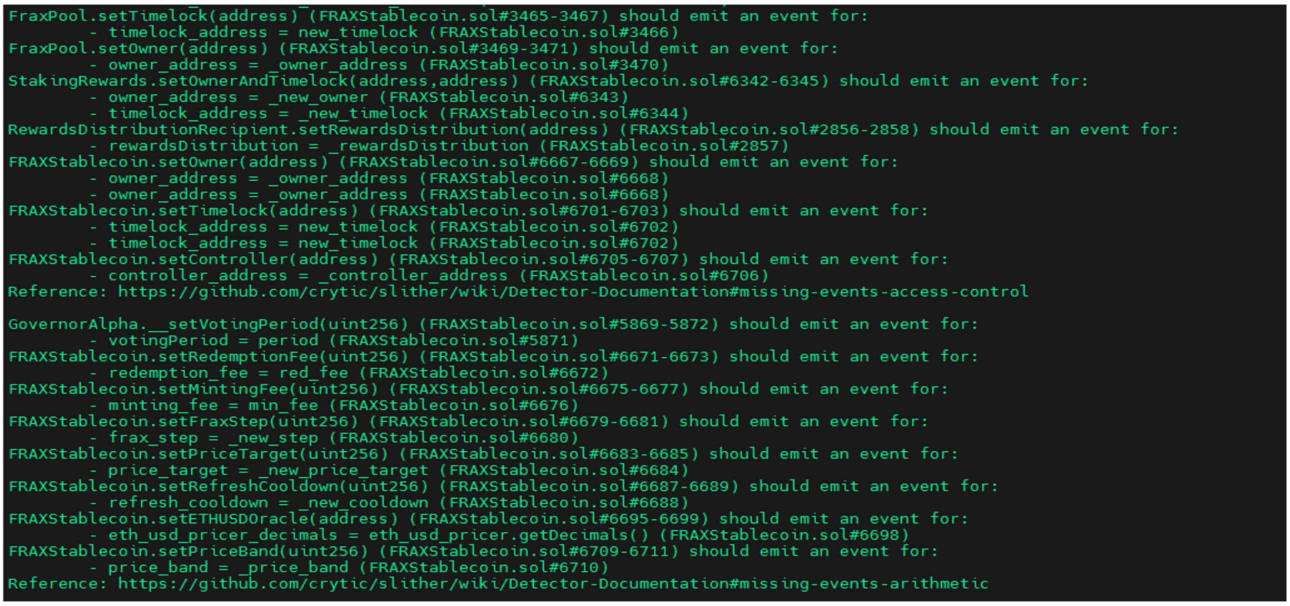


Slither Results Log

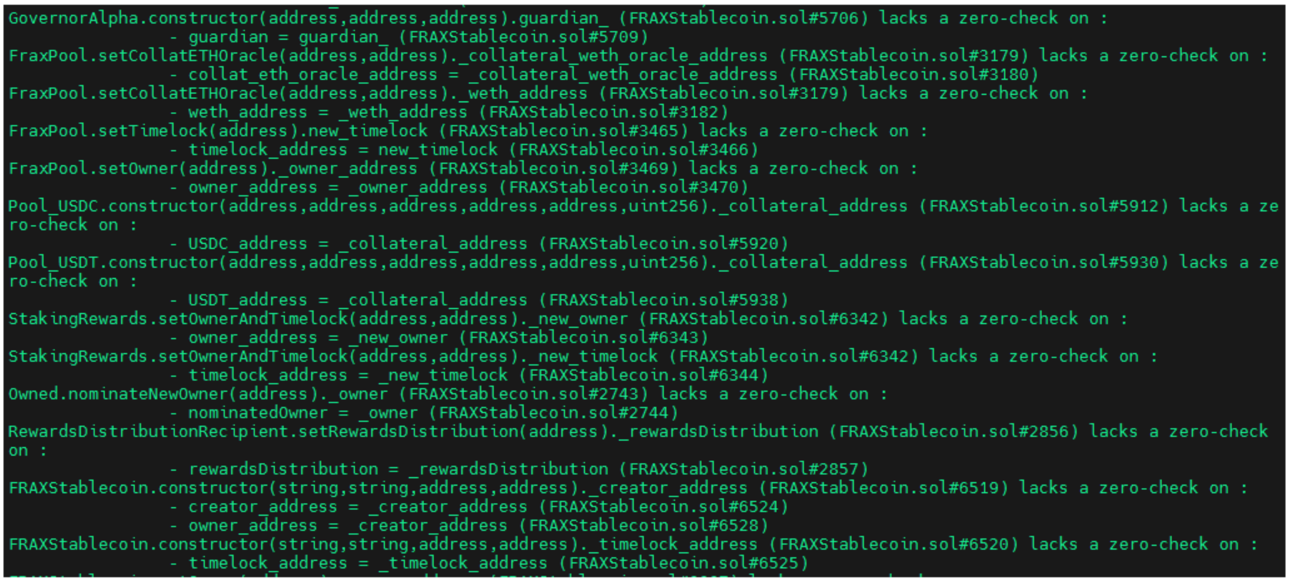
Slither is a Solidity static analysis framework that uses vulnerability detectors, displays contract details, and provides an API for writing custom analyses. It helps developers identify vulnerabilities, improve code comprehension, and prototype custom analyses quickly. The analysis includes a report with warnings and errors, allowing developers to quickly prototype and fix issues.

We did the analysis of the project altogether. Below are the results.

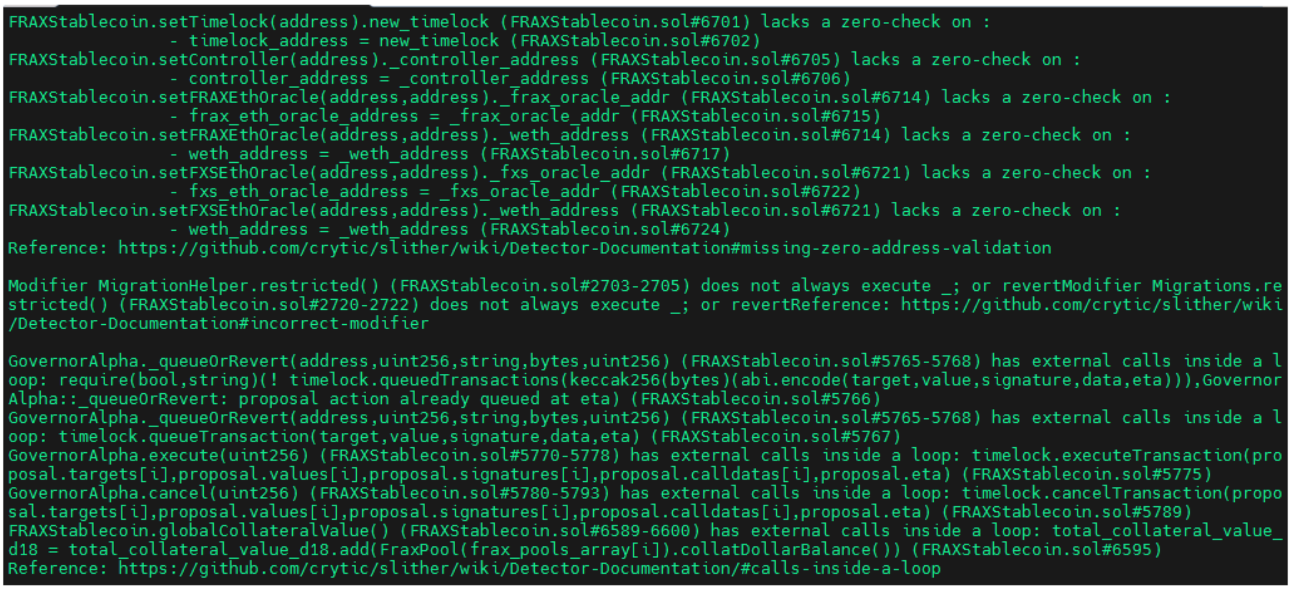
Slither Log >> FRAXStablecoin.sol



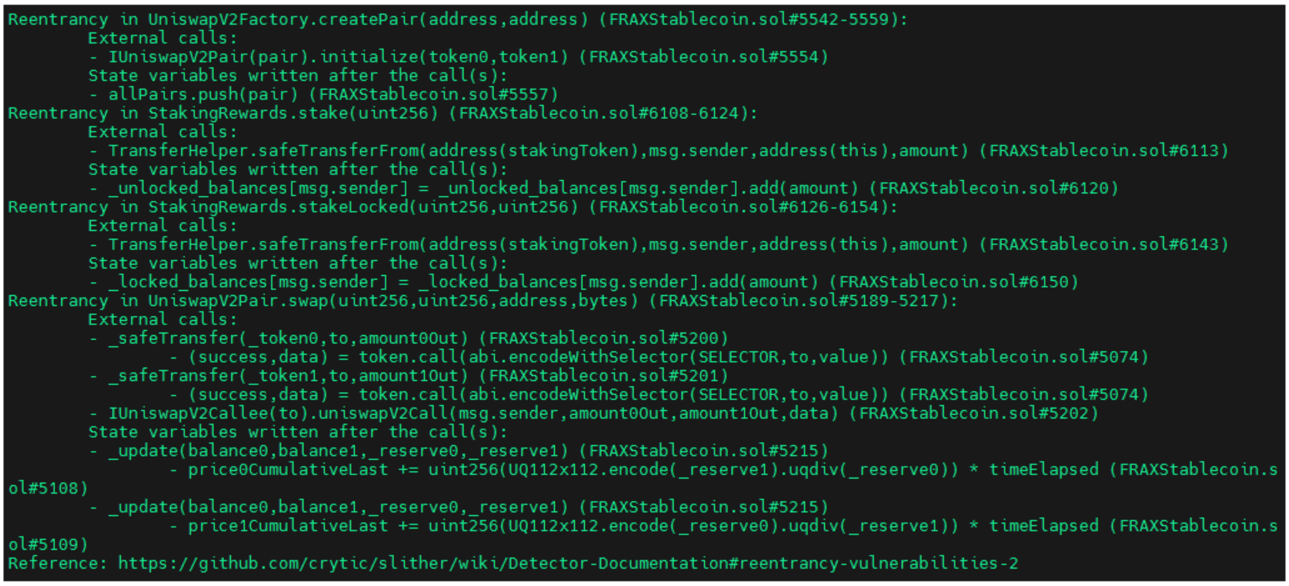
```
FraxPool.setTimelock(address) (FRAXStablecoin.sol#3465-3467) should emit an event for:
        - timelock_address = new_timelock (FRAXStablecoin.sol#3466)
FraxPool.setOwner(address) (FRAXStablecoin.sol#3469-3471) should emit an event for:
        - owner_address = _owner_address (FRAXStablecoin.sol#3470)
StakingRewards.setOwnerAndTimelock(address,address) (FRAXStablecoin.sol#6342-6345) should emit an event for:
        - owner_address = _new_owner (FRAXStablecoin.sol#6343)
        - timelock_address = _new_timelock (FRAXStablecoin.sol#6344)
RewardsDistributionRecipient.setRewardsDistribution(address) (FRAXStablecoin.sol#2856-2858) should emit an event for:
        - rewardsDistribution = _rewardsDistribution (FRAXStablecoin.sol#2857)
FRAXStablecoin.setOwner(address) (FRAXStablecoin.sol#6667-6669) should emit an event for:
        - owner_address = _owner_address (FRAXStablecoin.sol#6668)
        - owner_address = _owner_address (FRAXStablecoin.sol#6668)
FRAXStablecoin.setTimelock(address) (FRAXStablecoin.sol#6701-6703) should emit an event for:
        - timelock_address = new_timelock (FRAXStablecoin.sol#6702)
        - timelock_address = new_timelock (FRAXStablecoin.sol#6702)
FRAXStablecoin.setController(address) (FRAXStablecoin.sol#6705-6707) should emit an event for:
        - controller_address = _controller_address (FRAXStablecoin.sol#6706)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

GovernorAlpha.__setVotingPeriod(uint256) (FRAXStablecoin.sol#5869-5872) should emit an event for:
        - votingPeriod = period (FRAXStablecoin.sol#5871)
FRAXStablecoin.setRedemptionFee(uint256) (FRAXStablecoin.sol#6671-6673) should emit an event for:
        - redemption_fee = red_fee (FRAXStablecoin.sol#6672)
FRAXStablecoin.setMintingFee(uint256) (FRAXStablecoin.sol#6675-6677) should emit an event for:
        - minting_fee = min_fee (FRAXStablecoin.sol#6676)
FRAXStablecoin.setFraxStep(uint256) (FRAXStablecoin.sol#6679-6681) should emit an event for:
        - frax_step = _new_step (FRAXStablecoin.sol#6680)
FRAXStablecoin.setPriceTarget(uint256) (FRAXStablecoin.sol#6683-6685) should emit an event for:
        - price_target = _new_price_target (FRAXStablecoin.sol#6684)
FRAXStablecoin.setRefreshCooldown(uint256) (FRAXStablecoin.sol#6687-6689) should emit an event for:
        - refresh_cooldown = _new_cooldown (FRAXStablecoin.sol#6688)
FRAXStablecoin.setETHUSDOracle(address) (FRAXStablecoin.sol#6695-6699) should emit an event for:
        - eth_usd_pricer_decimals = eth_usd_pricer.getDecimals() (FRAXStablecoin.sol#6698)
FRAXStablecoin.setPriceBand(uint256) (FRAXStablecoin.sol#6709-6711) should emit an event for:
        - price_band = _price_band (FRAXStablecoin.sol#6710)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
```



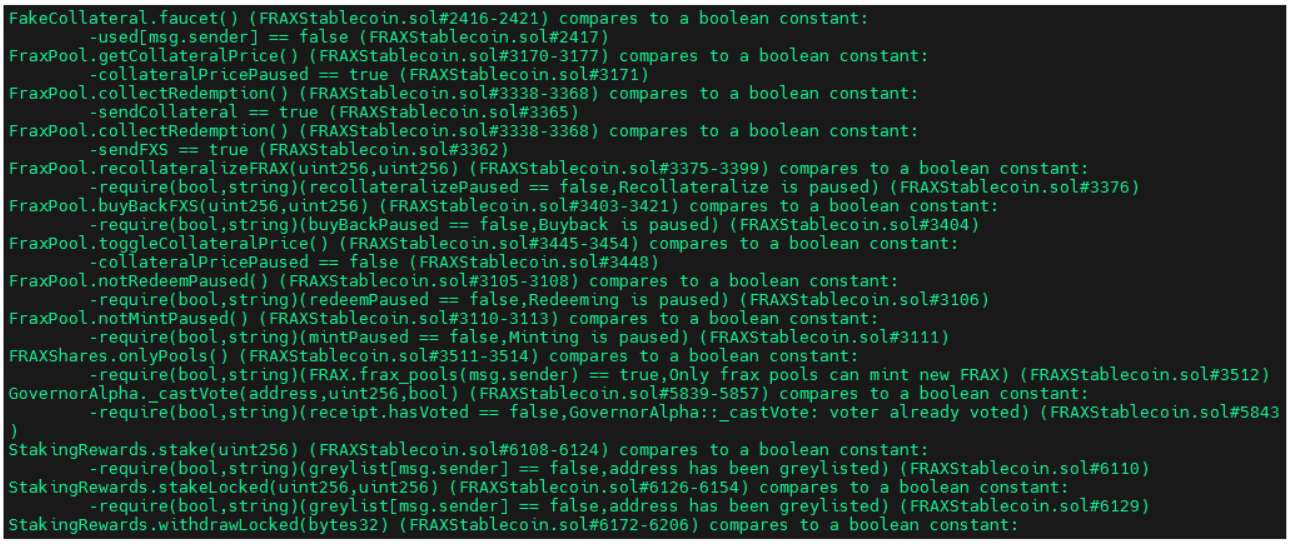
```
GovernorAlpha.constructor(address,address,address).guardian_ (FRAXStablecoin.sol#5706) lacks a zero-check on :
                - guardian = guardian_ (FRAXStablecoin.sol#5709)
FraxPool.setCollatETHOracle(address,address)._collateral_weth_oracle_address (FRAXStablecoin.sol#3179) lacks a zero-check on :
                - collat_eth_oracle_address = _collateral_weth_oracle_address (FRAXStablecoin.sol#3180)
FraxPool.setCollatETHOracle(address,address)._weth_address (FRAXStablecoin.sol#3179) lacks a zero-check on :
                - weth_address = _weth_address (FRAXStablecoin.sol#3182)
FraxPool.setTimelock(address).new_timelock (FRAXStablecoin.sol#3465) lacks a zero-check on :
                - timelock_address = new_timelock (FRAXStablecoin.sol#3466)
FraxPool.setOwner(address)._owner_address (FRAXStablecoin.sol#3469) lacks a zero-check on :
                - owner_address = _owner_address (FRAXStablecoin.sol#3470)
Pool_USDC.constructor(address,address,address,address,address,uint256)._collateral_address (FRAXStablecoin.sol#5912) lacks a ze
ro-check on :
                - USDC_address = _collateral_address (FRAXStablecoin.sol#5920)
Pool_USDT.constructor(address,address,address,address,address,uint256)._collateral_address (FRAXStablecoin.sol#5930) lacks a ze
ro-check on :
                - USDT_address = _collateral_address (FRAXStablecoin.sol#5938)
StakingRewards.setOwnerAndTimelock(address,address)._new_owner (FRAXStablecoin.sol#6342) lacks a zero-check on :
                - owner_address = _new_owner (FRAXStablecoin.sol#6343)
StakingRewards.setOwnerAndTimelock(address,address)._new_timelock (FRAXStablecoin.sol#6342) lacks a zero-check on :
                - timelock_address = _new_timelock (FRAXStablecoin.sol#6344)
Owned.nominateNewOwner(address)._owner (FRAXStablecoin.sol#2743) lacks a zero-check on :
                - nominatedOwner = _owner (FRAXStablecoin.sol#2744)
RewardsDistributionRecipient.setRewardsDistribution(address)._rewardsDistribution (FRAXStablecoin.sol#2856) lacks a zero-check
on :
                - rewardsDistribution = _rewardsDistribution (FRAXStablecoin.sol#2857)
FRAXStablecoin.constructor(string,string,address,address)._creator_address (FRAXStablecoin.sol#6519) lacks a zero-check on :
                - creator_address = _creator_address (FRAXStablecoin.sol#6524)
                - owner_address = _creator_address (FRAXStablecoin.sol#6528)
FRAXStablecoin.constructor(string,string,address,address)._timelock_address (FRAXStablecoin.sol#6520) lacks a zero-check on :
                - timelock_address = _timelock_address (FRAXStablecoin.sol#6525)
```



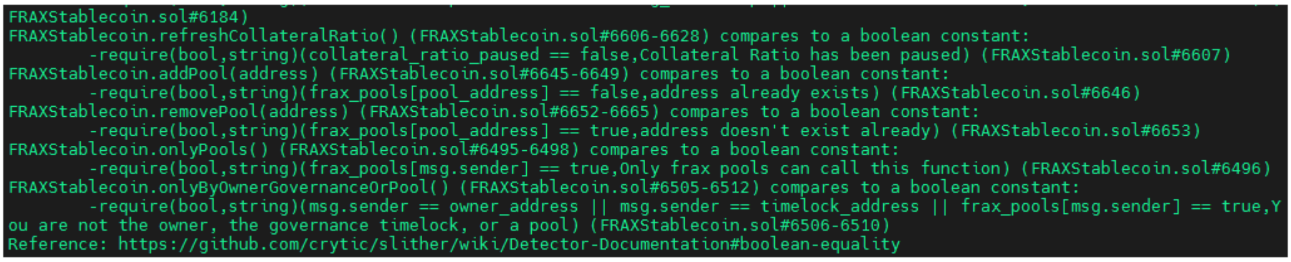
```
FRAXStablecoin.setTimelock(address).new_timelock (FRAXStablecoin.sol#6701) lacks a zero-check on :
                - timelock_address = new_timelock (FRAXStablecoin.sol#6702)
FRAXStablecoin.setController(address)._controller_address (FRAXStablecoin.sol#6705) lacks a zero-check on :
                - controller_address = _controller_address (FRAXStablecoin.sol#6706)
FRAXStablecoin.setFRAXEthOracle(address,address)._frax_oracle_addr (FRAXStablecoin.sol#6714) lacks a zero-check on :
                - frax_eth_oracle_address = _frax_oracle_addr (FRAXStablecoin.sol#6715)
FRAXStablecoin.setFRAXEthOracle(address,address)._weth_address (FRAXStablecoin.sol#6714) lacks a zero-check on :
                - weth_address = _weth_address (FRAXStablecoin.sol#6717)
FRAXStablecoin.setFXSEthOracle(address,address)._fxs_oracle_addr (FRAXStablecoin.sol#6721) lacks a zero-check on :
                - fxs_eth_oracle_address = _fxs_oracle_addr (FRAXStablecoin.sol#6722)
FRAXStablecoin.setFXSEthOracle(address,address)._weth_address (FRAXStablecoin.sol#6721) lacks a zero-check on :
                - weth_address = _weth_address (FRAXStablecoin.sol#6724)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Modifier MigrationHelper.restricted() (FRAXStablecoin.sol#2703-2705) does not always execute _; or revertModifier Migrations.re
stricted() (FRAXStablecoin.sol#2720-2722) does not always execute _; or revertReference: https://github.com/crytic/slither/wiki
/Detector-Documentation#incorrect-modifier

GovernorAlpha._queueOrRevert(address,uint256,string,bytes,uint256) (FRAXStablecoin.sol#5765-5768) has external calls inside a l
oop: require(bool,string)(! timelock.queuedTransactions(keccak256(bytes)(abi.encode(target,value,signature,data,eta))),Governor
Alpha::_queueOrRevert: proposal action already queued at eta) (FRAXStablecoin.sol#5766)
GovernorAlpha._queueOrRevert(address,uint256,string,bytes,uint256) (FRAXStablecoin.sol#5765-5768) has external calls inside a l
oop: timelock.queueTransaction(target,value,signature,data,eta) (FRAXStablecoin.sol#5767)
GovernorAlpha.execute(uint256) (FRAXStablecoin.sol#5770-5778) has external calls inside a loop: timelock.executeTransaction(pro
posal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (FRAXStablecoin.sol#5775)
GovernorAlpha.cancel(uint256) (FRAXStablecoin.sol#5780-5793) has external calls inside a loop: timelock.cancelTransaction(propo
sal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (FRAXStablecoin.sol#5789)
FRAXStablecoin.globalCollateralValue() (FRAXStablecoin.sol#6589-6600) has external calls inside a loop: total_collateral_value_
d18 = total_collateral_value_d18.add(FraxPool(frax_pools_array[i]).collatDollarBalance()) (FRAXStablecoin.sol#6595)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```



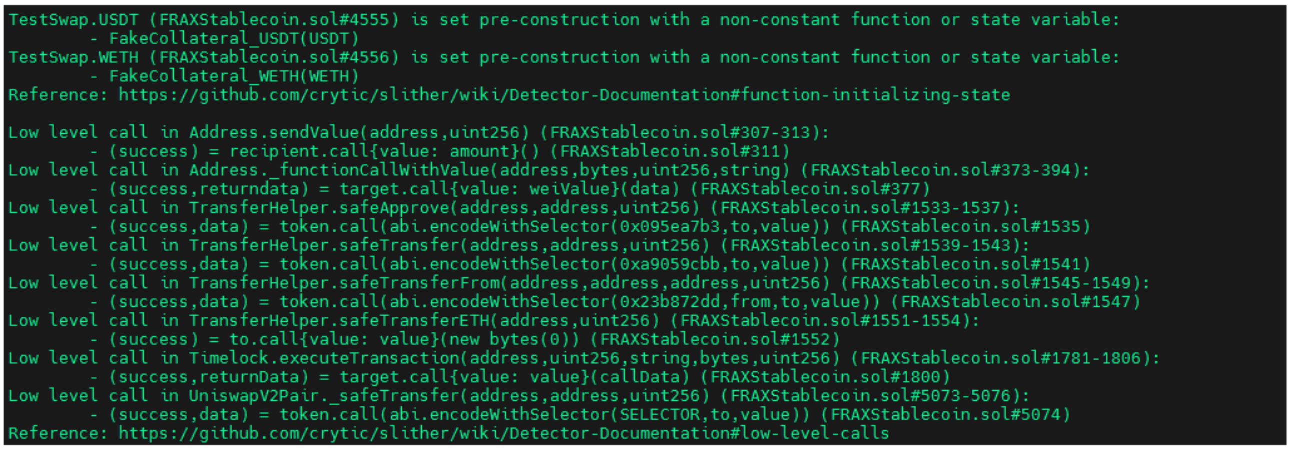
```
Reentrancy in UniswapV2Factory.createPair(address,address) (FRAXStablecoin.sol#5542-5559):
        External calls:
        - IUniswapV2Pair(pair).initialize(token0,token1) (FRAXStablecoin.sol#5554)
        State variables written after the call(s):
        - allPairs.push(pair) (FRAXStablecoin.sol#5557)
Reentrancy in StakingRewards.stake(uint256) (FRAXStablecoin.sol#6108-6124):
        External calls:
        - TransferHelper.safeTransferFrom(address(stakingToken),msg.sender,address(this),amount) (FRAXStablecoin.sol#6113)
        State variables written after the call(s):
        - _unlocked_balances[msg.sender] = _unlocked_balances[msg.sender].add(amount) (FRAXStablecoin.sol#6120)
Reentrancy in StakingRewards.stakeLocked(uint256,uint256) (FRAXStablecoin.sol#6126-6154):
        External calls:
        - TransferHelper.safeTransferFrom(address(stakingToken),msg.sender,address(this),amount) (FRAXStablecoin.sol#6143)
        State variables written after the call(s):
        - _locked_balances[msg.sender] = _locked_balances[msg.sender].add(amount) (FRAXStablecoin.sol#6150)
Reentrancy in UniswapV2Pair.swap(uint256,uint256,address,bytes) (FRAXStablecoin.sol#5189-5217):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (FRAXStablecoin.sol#5200)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (FRAXStablecoin.sol#5074)
        - _safeTransfer(_token1,to,amount1Out) (FRAXStablecoin.sol#5201)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (FRAXStablecoin.sol#5074)
        - IUniswapV2Callee(to).uniswapV2Call(msg.sender,amount0Out,amount1Out,data) (FRAXStablecoin.sol#5202)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (FRAXStablecoin.sol#5215)
                - price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (FRAXStablecoin.s
ol#5108)
        - _update(balance0,balance1,_reserve0,_reserve1) (FRAXStablecoin.sol#5215)
                - price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (FRAXStablecoin.s
ol#5109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```



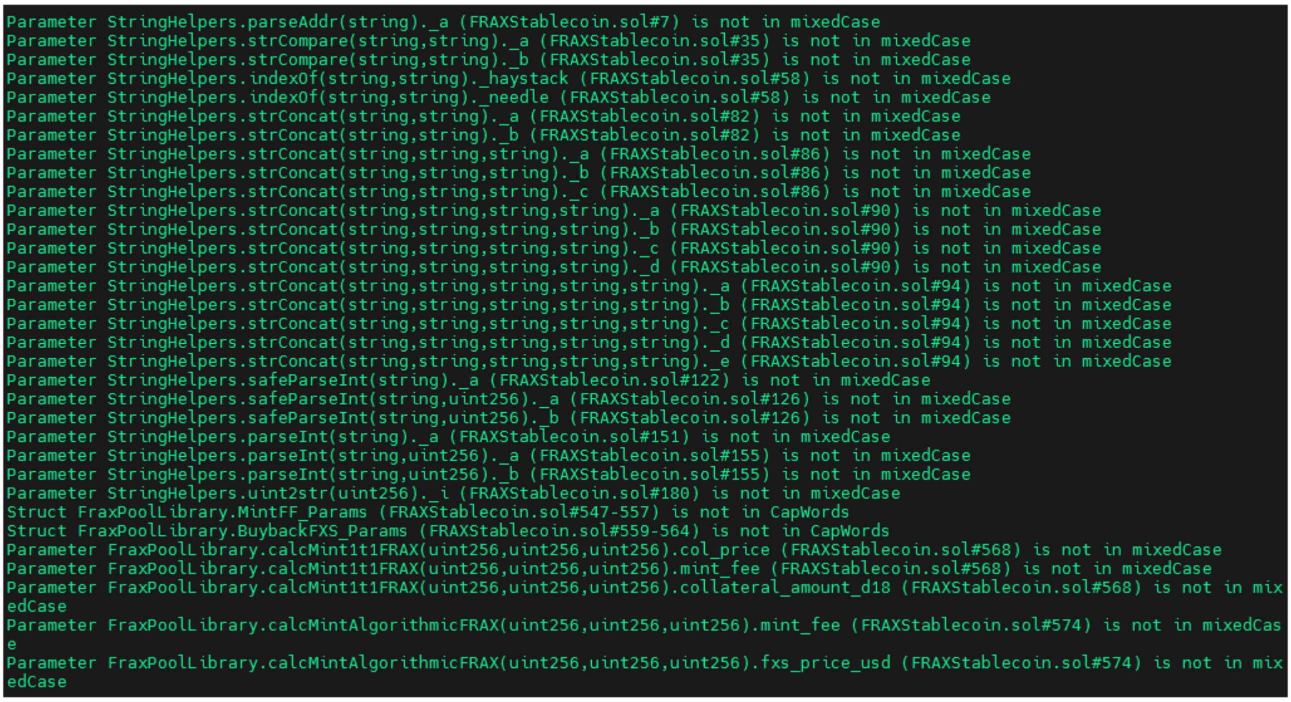
```
FakeCollateral.faucet() (FRAXStablecoin.sol#2416-2421) compares to a boolean constant:
        -used[msg.sender] == false (FRAXStablecoin.sol#2417)
FraxPool.getCollateralPrice() (FRAXStablecoin.sol#3170-3177) compares to a boolean constant:
        -collateralPricePaused == true (FRAXStablecoin.sol#3171)
FraxPool.collectRedemption() (FRAXStablecoin.sol#3338-3368) compares to a boolean constant:
        -sendCollateral == true (FRAXStablecoin.sol#3365)
FraxPool.collectRedemption() (FRAXStablecoin.sol#3338-3368) compares to a boolean constant:
        -sendFXS == true (FRAXStablecoin.sol#3362)
FraxPool.recollateralizeFRAX(uint256,uint256) (FRAXStablecoin.sol#3375-3399) compares to a boolean constant:
        -require(bool,string)(recollateralizePaused == false,Recollateralize is paused) (FRAXStablecoin.sol#3376)
FraxPool.buyBackFXS(uint256,uint256) (FRAXStablecoin.sol#3403-3421) compares to a boolean constant:
        -require(bool,string)(buyBackPaused == false,Buyback is paused) (FRAXStablecoin.sol#3404)
FraxPool.toggleCollateralPrice() (FRAXStablecoin.sol#3445-3454) compares to a boolean constant:
        -collateralPricePaused == false (FRAXStablecoin.sol#3448)
FraxPool.notRedeemPaused() (FRAXStablecoin.sol#3105-3108) compares to a boolean constant:
        -require(bool,string)(redeemPaused == false,Redeeming is paused) (FRAXStablecoin.sol#3106)
FraxPool.notMintPaused() (FRAXStablecoin.sol#3110-3113) compares to a boolean constant:
        -require(bool,string)(mintPaused == false,Minting is paused) (FRAXStablecoin.sol#3111)
FRAXShares.onlyPools() (FRAXStablecoin.sol#3511-3514) compares to a boolean constant:
        -require(bool,string)(FRAX.frax_pools(msg.sender) == true,Only frax pools can mint new FRAX) (FRAXStablecoin.sol#3512)
GovernorAlpha._castVote(address,uint256,bool) (FRAXStablecoin.sol#5839-5857) compares to a boolean constant:
        -require(bool,string)(receipt.hasVoted == false,GovernorAlpha::_castVote: voter already voted) (FRAXStablecoin.sol#5843
)
StakingRewards.stake(uint256) (FRAXStablecoin.sol#6108-6124) compares to a boolean constant:
        -require(bool,string)(greylist[msg.sender] == false,address has been greylisted) (FRAXStablecoin.sol#6110)
StakingRewards.stakeLocked(uint256,uint256) (FRAXStablecoin.sol#6126-6154) compares to a boolean constant:
        -require(bool,string)(greylist[msg.sender] == false,address has been greylisted) (FRAXStablecoin.sol#6129)
StakingRewards.withdrawLocked(bytes32) (FRAXStablecoin.sol#6172-6206) compares to a boolean constant:
```



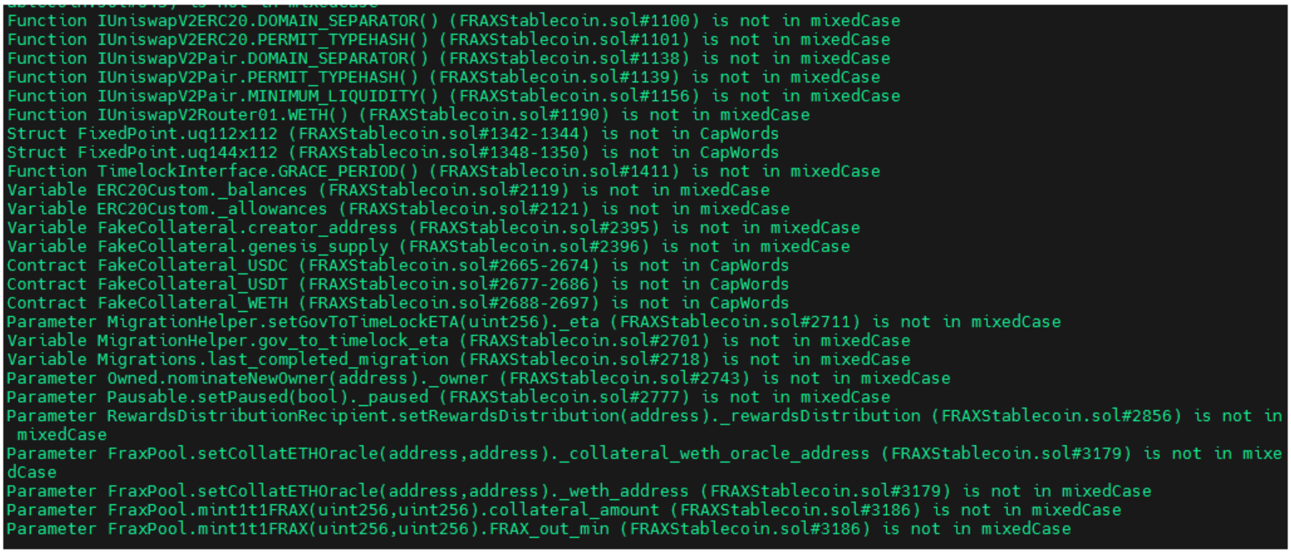
```
FRAXStablecoin.sol#6184)
FRAXStablecoin.refreshCollateralRatio() (FRAXStablecoin.sol#6606-6628) compares to a boolean constant:
        -require(bool,string)(collateral_ratio_paused == false,Collateral Ratio has been paused) (FRAXStablecoin.sol#6607)
FRAXStablecoin.addPool(address) (FRAXStablecoin.sol#6645-6649) compares to a boolean constant:
        -require(bool,string)(frax_pools[pool_address] == false,address already exists) (FRAXStablecoin.sol#6646)
FRAXStablecoin.removePool(address) (FRAXStablecoin.sol#6652-6665) compares to a boolean constant:
        -require(bool,string)(frax_pools[pool_address] == true,address doesn't exist already) (FRAXStablecoin.sol#6653)
FRAXStablecoin.onlyPools() (FRAXStablecoin.sol#6495-6498) compares to a boolean constant:
        -require(bool,string)(frax_pools[msg.sender] == true,Only frax pools can call this function) (FRAXStablecoin.sol#6496)
FRAXStablecoin.onlyByOwnerGovernanceOrPool() (FRAXStablecoin.sol#6505-6512) compares to a boolean constant:
        -require(bool,string)(msg.sender == owner_address || msg.sender == timelock_address || frax_pools[msg.sender] == true,Y
ou are not the owner, the governance timelock, or a pool) (FRAXStablecoin.sol#6506-6510)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```



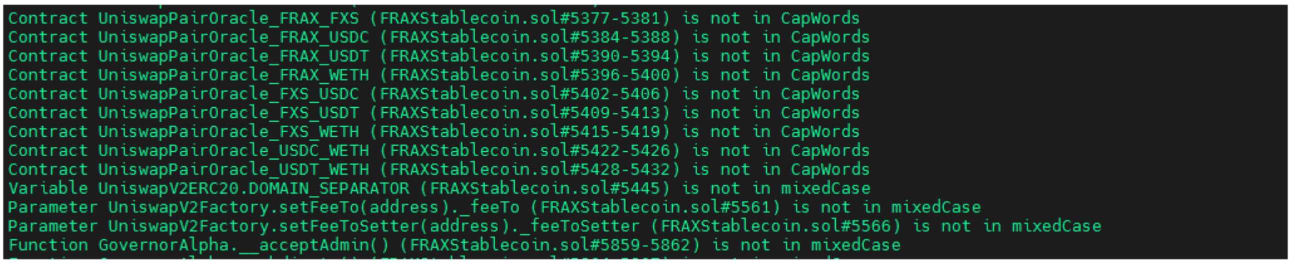
```
TestSwap.USDT (FRAXStablecoin.sol#4555) is set pre-construction with a non-constant function or state variable:
        - FakeCollateral_USDT(USDT)
TestSwap.WETH (FRAXStablecoin.sol#4556) is set pre-construction with a non-constant function or state variable:
        - FakeCollateral_WETH(WETH)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state

Low level call in Address.sendValue(address,uint256) (FRAXStablecoin.sol#307-313):
        - (success) = recipient.call{value: amount}() (FRAXStablecoin.sol#311)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (FRAXStablecoin.sol#373-394):
        - (success,returndata) = target.call{value: weiValue}(data) (FRAXStablecoin.sol#377)
Low level call in TransferHelper.safeApprove(address,address,uint256) (FRAXStablecoin.sol#1533-1537):
        - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (FRAXStablecoin.sol#1535)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (FRAXStablecoin.sol#1539-1543):
        - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (FRAXStablecoin.sol#1541)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (FRAXStablecoin.sol#1545-1549):
        - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (FRAXStablecoin.sol#1547)
Low level call in TransferHelper.safeTransferETH(address,uint256) (FRAXStablecoin.sol#1551-1554):
        - (success) = to.call{value: value}(new bytes(0)) (FRAXStablecoin.sol#1552)
Low level call in Timelock.executeTransaction(address,uint256,string,bytes,uint256) (FRAXStablecoin.sol#1781-1806):
        - (success,returnData) = target.call{value: value}(callData) (FRAXStablecoin.sol#1800)
Low level call in UniswapV2Pair._safeTransfer(address,address,uint256) (FRAXStablecoin.sol#5073-5076):
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (FRAXStablecoin.sol#5074)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```



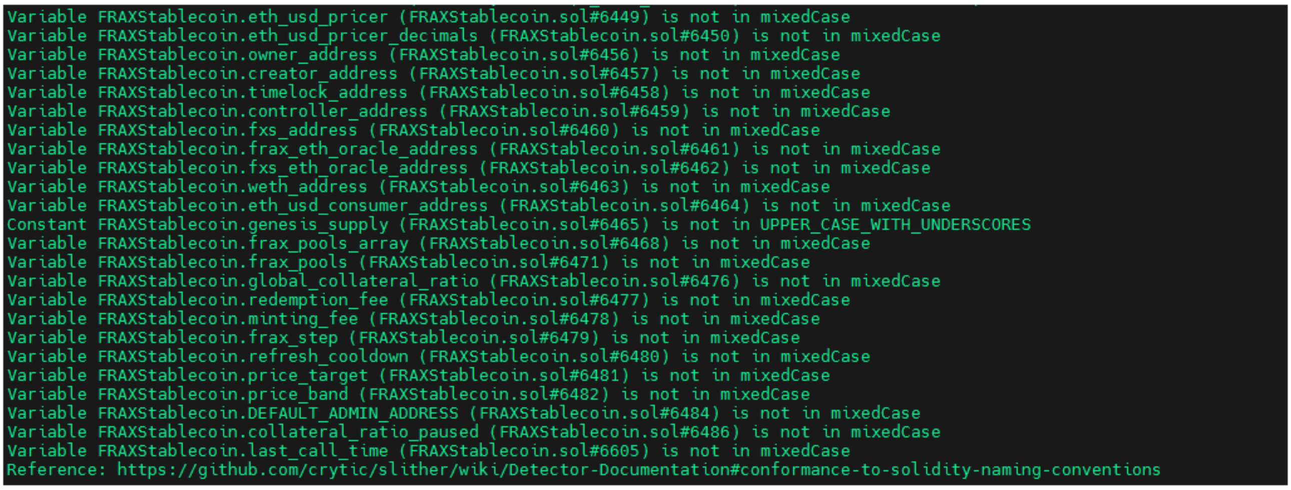
```
Parameter StringHelpers.parseAddr(string)._a (FRAXStablecoin.sol#7) is not in mixedCase
Parameter StringHelpers.strCompare(string,string)._a (FRAXStablecoin.sol#35) is not in mixedCase
Parameter StringHelpers.strCompare(string,string)._b (FRAXStablecoin.sol#35) is not in mixedCase
Parameter StringHelpers.indexOf(string,string)._haystack (FRAXStablecoin.sol#58) is not in mixedCase
Parameter StringHelpers.indexOf(string,string)._needle (FRAXStablecoin.sol#58) is not in mixedCase
Parameter StringHelpers.strConcat(string,string)._a (FRAXStablecoin.sol#82) is not in mixedCase
Parameter StringHelpers.strConcat(string,string)._b (FRAXStablecoin.sol#82) is not in mixedCase
Parameter StringHelpers.strConcat(string,string,string)._a (FRAXStablecoin.sol#86) is not in mixedCase
Parameter StringHelpers.strConcat(string,string,string)._b (FRAXStablecoin.sol#86) is not in mixedCase
Parameter StringHelpers.strConcat(string,string,string)._c (FRAXStablecoin.sol#86) is not in mixedCase
Parameter StringHelpers.strConcat(string,string,string,string)._a (FRAXStablecoin.sol#90) is not in mixedCase
Parameter StringHelpers.strConcat(string,string,string,string)._b (FRAXStablecoin.sol#90) is not in mixedCase
Parameter StringHelpers.strConcat(string,string,string,string)._c (FRAXStablecoin.sol#90) is not in mixedCase
Parameter StringHelpers.strConcat(string,string,string,string)._d (FRAXStablecoin.sol#90) is not in mixedCase
Parameter StringHelpers.strConcat(string,string,string,string,string)._a (FRAXStablecoin.sol#94) is not in mixedCase
Parameter StringHelpers.strConcat(string,string,string,string,string)._b (FRAXStablecoin.sol#94) is not in mixedCase
Parameter StringHelpers.strConcat(string,string,string,string,string)._c (FRAXStablecoin.sol#94) is not in mixedCase
Parameter StringHelpers.strConcat(string,string,string,string,string)._d (FRAXStablecoin.sol#94) is not in mixedCase
Parameter StringHelpers.strConcat(string,string,string,string,string)._e (FRAXStablecoin.sol#94) is not in mixedCase
Parameter StringHelpers.safeParseInt(string)._a (FRAXStablecoin.sol#122) is not in mixedCase
Parameter StringHelpers.safeParseInt(string,uint256)._a (FRAXStablecoin.sol#126) is not in mixedCase
Parameter StringHelpers.safeParseInt(string,uint256)._b (FRAXStablecoin.sol#126) is not in mixedCase
Parameter StringHelpers.parseInt(string)._a (FRAXStablecoin.sol#151) is not in mixedCase
Parameter StringHelpers.parseInt(string,uint256)._a (FRAXStablecoin.sol#155) is not in mixedCase
Parameter StringHelpers.parseInt(string,uint256)._b (FRAXStablecoin.sol#155) is not in mixedCase
Parameter StringHelpers.uint2str(uint256)._i (FRAXStablecoin.sol#180) is not in mixedCase
Struct FraxPoolLibrary.MintFF_Params (FRAXStablecoin.sol#547-557) is not in CapWords
Struct FraxPoolLibrary.BuybackFXS_Params (FRAXStablecoin.sol#559-564) is not in CapWords
Parameter FraxPoolLibrary.calcMint1t1FRAX(uint256,uint256,uint256).col_price (FRAXStablecoin.sol#568) is not in mixedCase
Parameter FraxPoolLibrary.calcMint1t1FRAX(uint256,uint256,uint256).mint_fee (FRAXStablecoin.sol#568) is not in mixedCase
Parameter FraxPoolLibrary.calcMint1t1FRAX(uint256,uint256,uint256).collateral_amount_d18 (FRAXStablecoin.sol#568) is not in mix
edCase
Parameter FraxPoolLibrary.calcMintAlgorithmicFRAX(uint256,uint256,uint256).mint_fee (FRAXStablecoin.sol#574) is not in mixedCas
e
Parameter FraxPoolLibrary.calcMintAlgorithmicFRAX(uint256,uint256,uint256).fxs_price_usd (FRAXStablecoin.sol#574) is not in mix
edCase
```



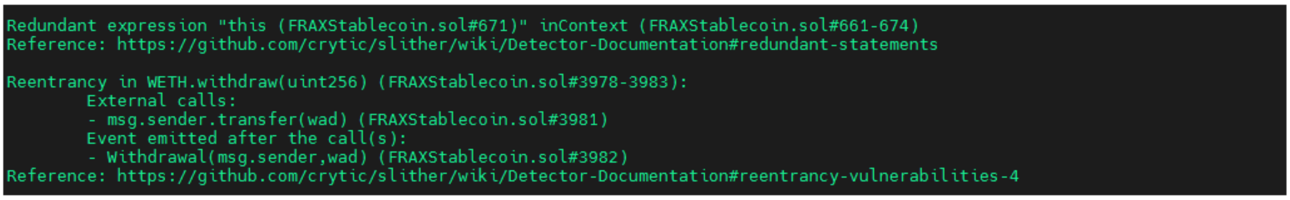
```
Function IUniswapV2ERC20.DOMAIN_SEPARATOR() (FRAXStablecoin.sol#1100) is not in mixedCase
Function IUniswapV2ERC20.PERMIT_TYPEHASH() (FRAXStablecoin.sol#1101) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (FRAXStablecoin.sol#1138) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (FRAXStablecoin.sol#1139) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (FRAXStablecoin.sol#1156) is not in mixedCase
Function IUniswapV2Router01.WETH() (FRAXStablecoin.sol#1190) is not in mixedCase
Struct FixedPoint.uq112x112 (FRAXStablecoin.sol#1342-1344) is not in CapWords
Struct FixedPoint.uq144x112 (FRAXStablecoin.sol#1348-1350) is not in CapWords
Function TimelockInterface.GRACE_PERIOD() (FRAXStablecoin.sol#1411) is not in mixedCase
Variable ERC20Custom._balances (FRAXStablecoin.sol#2119) is not in mixedCase
Variable ERC20Custom._allowances (FRAXStablecoin.sol#2121) is not in mixedCase
Variable FakeCollateral.creator_address (FRAXStablecoin.sol#2395) is not in mixedCase
Variable FakeCollateral.genesis_supply (FRAXStablecoin.sol#2396) is not in mixedCase
Contract FakeCollateral_USDC (FRAXStablecoin.sol#2665-2674) is not in CapWords
Contract FakeCollateral_USDT (FRAXStablecoin.sol#2677-2686) is not in CapWords
Contract FakeCollateral_WETH (FRAXStablecoin.sol#2688-2697) is not in CapWords
Parameter MigrationHelper.setGovToTimeLockETA(uint256)._eta (FRAXStablecoin.sol#2711) is not in mixedCase
Variable MigrationHelper.gov_to_timelock_eta (FRAXStablecoin.sol#2701) is not in mixedCase
Variable Migrations.last_completed_migration (FRAXStablecoin.sol#2718) is not in mixedCase
Parameter Owned.nominateNewOwner(address)._owner (FRAXStablecoin.sol#2743) is not in mixedCase
Parameter Pausable.setPaused(bool)._paused (FRAXStablecoin.sol#2777) is not in mixedCase
Parameter RewardsDistributionRecipient.setRewardsDistribution(address)._rewardsDistribution (FRAXStablecoin.sol#2856) is not in
 mixedCase
Parameter FraxPool.setCollatETHOracle(address,address)._collateral_weth_oracle_address (FRAXStablecoin.sol#3179) is not in mixe
dCase
Parameter FraxPool.setCollatETHOracle(address,address)._weth_address (FRAXStablecoin.sol#3179) is not in mixedCase
Parameter FraxPool.mint1t1FRAX(uint256,uint256).collateral_amount (FRAXStablecoin.sol#3186) is not in mixedCase
Parameter FraxPool.mint1t1FRAX(uint256,uint256).FRAX_out_min (FRAXStablecoin.sol#3186) is not in mixedCase
```



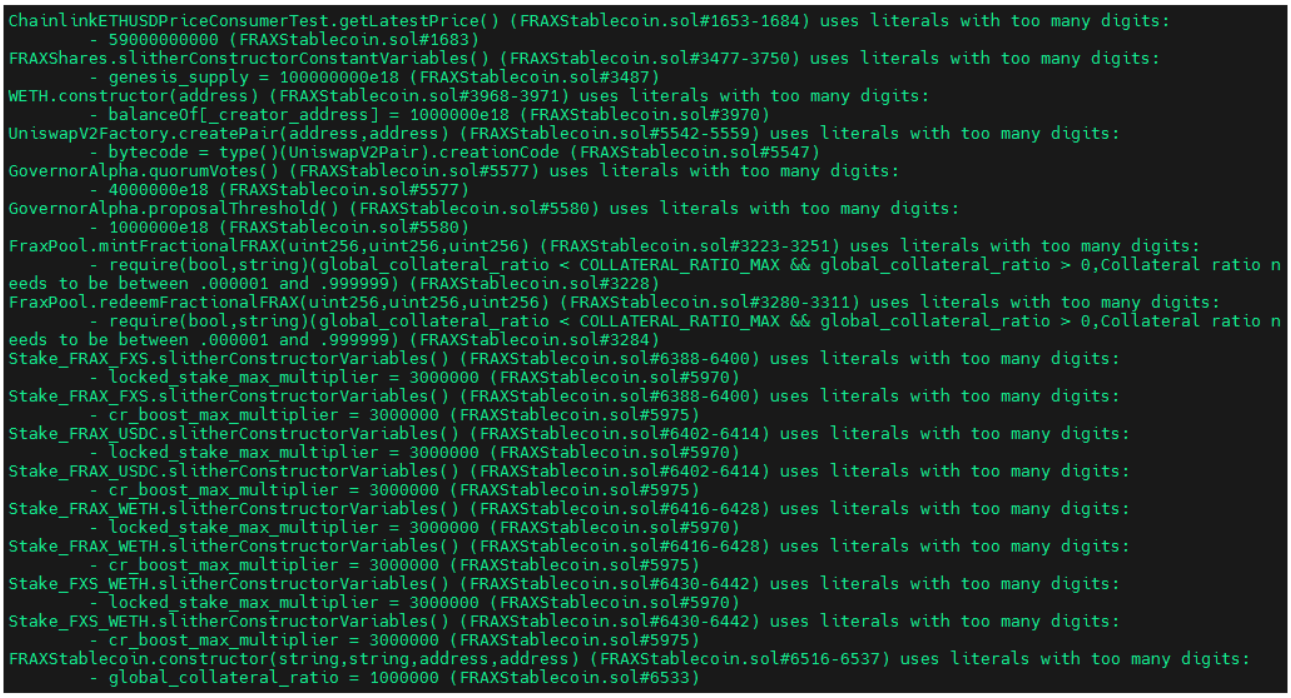
```
Contract UniswapPairOracle_FRAX_FXS (FRAXStablecoin.sol#5377-5381) is not in CapWords
Contract UniswapPairOracle_FRAX_USDC (FRAXStablecoin.sol#5384-5388) is not in CapWords
Contract UniswapPairOracle_FRAX_USDT (FRAXStablecoin.sol#5390-5394) is not in CapWords
Contract UniswapPairOracle_FRAX_WETH (FRAXStablecoin.sol#5396-5400) is not in CapWords
Contract UniswapPairOracle_FXS_USDC (FRAXStablecoin.sol#5402-5406) is not in CapWords
Contract UniswapPairOracle_FXS_USDT (FRAXStablecoin.sol#5409-5413) is not in CapWords
Contract UniswapPairOracle_FXS_WETH (FRAXStablecoin.sol#5415-5419) is not in CapWords
Contract UniswapPairOracle_USDC_WETH (FRAXStablecoin.sol#5422-5426) is not in CapWords
Contract UniswapPairOracle_USDT_WETH (FRAXStablecoin.sol#5428-5432) is not in CapWords
Variable UniswapV2ERC20.DOMAIN_SEPARATOR (FRAXStablecoin.sol#5445) is not in mixedCase
Parameter UniswapV2Factory.setFeeTo(address)._feeTo (FRAXStablecoin.sol#5561) is not in mixedCase
Parameter UniswapV2Factory.setFeeToSetter(address)._feeToSetter (FRAXStablecoin.sol#5566) is not in mixedCase
Function GovernorAlpha.__acceptAdmin() (FRAXStablecoin.sol#5859-5862) is not in mixedCase
```



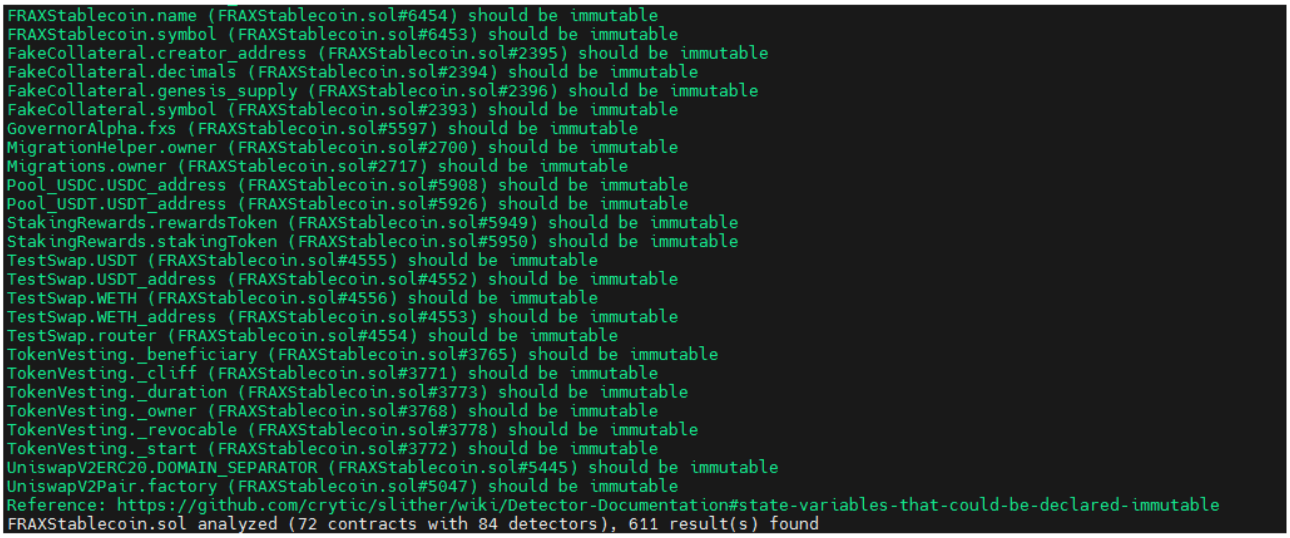
```
Variable FRAXStablecoin.eth_usd_pricer (FRAXStablecoin.sol#6449) is not in mixedCase
Variable FRAXStablecoin.eth_usd_pricer_decimals (FRAXStablecoin.sol#6450) is not in mixedCase
Variable FRAXStablecoin.owner_address (FRAXStablecoin.sol#6456) is not in mixedCase
Variable FRAXStablecoin.creator_address (FRAXStablecoin.sol#6457) is not in mixedCase
Variable FRAXStablecoin.timelock_address (FRAXStablecoin.sol#6458) is not in mixedCase
Variable FRAXStablecoin.controller_address (FRAXStablecoin.sol#6459) is not in mixedCase
Variable FRAXStablecoin.fxs_address (FRAXStablecoin.sol#6460) is not in mixedCase
Variable FRAXStablecoin.frax_eth_oracle_address (FRAXStablecoin.sol#6461) is not in mixedCase
Variable FRAXStablecoin.fxs_eth_oracle_address (FRAXStablecoin.sol#6462) is not in mixedCase
Variable FRAXStablecoin.weth_address (FRAXStablecoin.sol#6463) is not in mixedCase
Variable FRAXStablecoin.eth_usd_consumer_address (FRAXStablecoin.sol#6464) is not in mixedCase
Constant FRAXStablecoin.genesis_supply (FRAXStablecoin.sol#6465) is not in UPPER_CASE_WITH_UNDERSCORES
Variable FRAXStablecoin.frax_pools_array (FRAXStablecoin.sol#6468) is not in mixedCase
Variable FRAXStablecoin.frax_pools (FRAXStablecoin.sol#6471) is not in mixedCase
Variable FRAXStablecoin.global_collateral_ratio (FRAXStablecoin.sol#6476) is not in mixedCase
Variable FRAXStablecoin.redemption_fee (FRAXStablecoin.sol#6477) is not in mixedCase
Variable FRAXStablecoin.minting_fee (FRAXStablecoin.sol#6478) is not in mixedCase
Variable FRAXStablecoin.frax_step (FRAXStablecoin.sol#6479) is not in mixedCase
Variable FRAXStablecoin.refresh_cooldown (FRAXStablecoin.sol#6480) is not in mixedCase
Variable FRAXStablecoin.price_target (FRAXStablecoin.sol#6481) is not in mixedCase
Variable FRAXStablecoin.price_band (FRAXStablecoin.sol#6482) is not in mixedCase
Variable FRAXStablecoin.DEFAULT_ADMIN_ADDRESS (FRAXStablecoin.sol#6484) is not in mixedCase
Variable FRAXStablecoin.collateral_ratio_paused (FRAXStablecoin.sol#6486) is not in mixedCase
Variable FRAXStablecoin.last_call_time (FRAXStablecoin.sol#6605) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```



```
Redundant expression "this (FRAXStablecoin.sol#671)" inContext (FRAXStablecoin.sol#661-674)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Reentrancy in WETH.withdraw(uint256) (FRAXStablecoin.sol#3978-3983):
        External calls:
        - msg.sender.transfer(wad) (FRAXStablecoin.sol#3981)
        Event emitted after the call(s):
        - Withdrawal(msg.sender,wad) (FRAXStablecoin.sol#3982)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
```



```
ChainlinkETHUSDPriceConsumerTest.getLatestPrice() (FRAXStablecoin.sol#1653-1684) uses literals with too many digits:
        - 59000000000 (FRAXStablecoin.sol#1683)
FRAXShares.slitherConstructorConstantVariables() (FRAXStablecoin.sol#3477-3750) uses literals with too many digits:
        - genesis_supply = 100000000e18 (FRAXStablecoin.sol#3487)
WETH.constructor(address) (FRAXStablecoin.sol#3968-3971) uses literals with too many digits:
        - balanceOf[_creator_address] = 1000000e18 (FRAXStablecoin.sol#3970)
UniswapV2Factory.createPair(address,address) (FRAXStablecoin.sol#5542-5559) uses literals with too many digits:
        - bytecode = type()(UniswapV2Pair).creationCode (FRAXStablecoin.sol#5547)
GovernorAlpha.quorumVotes() (FRAXStablecoin.sol#5577) uses literals with too many digits:
        - 4000000e18 (FRAXStablecoin.sol#5577)
GovernorAlpha.proposalThreshold() (FRAXStablecoin.sol#5580) uses literals with too many digits:
        - 1000000e18 (FRAXStablecoin.sol#5580)
FraxPool.mintFractionalFRAX(uint256,uint256,uint256) (FRAXStablecoin.sol#3223-3251) uses literals with too many digits:
        - require(bool,string)(global_collateral_ratio < COLLATERAL_RATIO_MAX && global_collateral_ratio > 0,Collateral ratio n
eeds to be between .000001 and .999999) (FRAXStablecoin.sol#3228)
FraxPool.redeemFractionalFRAX(uint256,uint256,uint256) (FRAXStablecoin.sol#3280-3311) uses literals with too many digits:
        - require(bool,string)(global_collateral_ratio < COLLATERAL_RATIO_MAX && global_collateral_ratio > 0,Collateral ratio n
eeds to be between .000001 and .999999) (FRAXStablecoin.sol#3284)
Stake_FRAX_FXS.slitherConstructorVariables() (FRAXStablecoin.sol#6388-6400) uses literals with too many digits:
        - locked_stake_max_multiplier = 3000000 (FRAXStablecoin.sol#5970)
Stake_FRAX_FXS.slitherConstructorVariables() (FRAXStablecoin.sol#6388-6400) uses literals with too many digits:
        - cr_boost_max_multiplier = 3000000 (FRAXStablecoin.sol#5975)
Stake_FRAX_USDC.slitherConstructorVariables() (FRAXStablecoin.sol#6402-6414) uses literals with too many digits:
        - locked_stake_max_multiplier = 3000000 (FRAXStablecoin.sol#5970)
Stake_FRAX_USDC.slitherConstructorVariables() (FRAXStablecoin.sol#6402-6414) uses literals with too many digits:
        - cr_boost_max_multiplier = 3000000 (FRAXStablecoin.sol#5975)
Stake_FRAX_WETH.slitherConstructorVariables() (FRAXStablecoin.sol#6416-6428) uses literals with too many digits:
        - locked_stake_max_multiplier = 3000000 (FRAXStablecoin.sol#5970)
Stake_FRAX_WETH.slitherConstructorVariables() (FRAXStablecoin.sol#6416-6428) uses literals with too many digits:
        - cr_boost_max_multiplier = 3000000 (FRAXStablecoin.sol#5975)
Stake_FXS_WETH.slitherConstructorVariables() (FRAXStablecoin.sol#6430-6442) uses literals with too many digits:
        - locked_stake_max_multiplier = 3000000 (FRAXStablecoin.sol#5970)
Stake_FXS_WETH.slitherConstructorVariables() (FRAXStablecoin.sol#6430-6442) uses literals with too many digits:
        - cr_boost_max_multiplier = 3000000 (FRAXStablecoin.sol#5975)
FRAXStablecoin.constructor(string,string,address,address) (FRAXStablecoin.sol#6516-6537) uses literals with too many digits:
        - global_collateral_ratio = 1000000 (FRAXStablecoin.sol#6533)
```

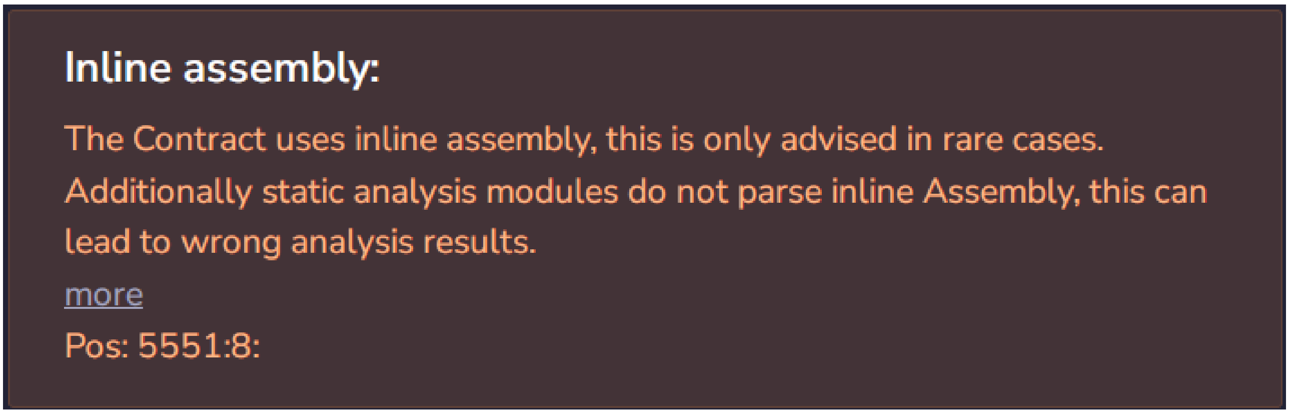


```
FRAXStablecoin.name (FRAXStablecoin.sol#6454) should be immutable
FRAXStablecoin.symbol (FRAXStablecoin.sol#6453) should be immutable
FakeCollateral.creator_address (FRAXStablecoin.sol#2395) should be immutable
FakeCollateral.decimals (FRAXStablecoin.sol#2394) should be immutable
FakeCollateral.genesis_supply (FRAXStablecoin.sol#2396) should be immutable
FakeCollateral.symbol (FRAXStablecoin.sol#2393) should be immutable
GovernorAlpha.fxs (FRAXStablecoin.sol#5597) should be immutable
MigrationHelper.owner (FRAXStablecoin.sol#2700) should be immutable
Migrations.owner (FRAXStablecoin.sol#2717) should be immutable
Pool_USDC.USDC_address (FRAXStablecoin.sol#5908) should be immutable
Pool_USDT.USDT_address (FRAXStablecoin.sol#5926) should be immutable
StakingRewards.rewardsToken (FRAXStablecoin.sol#5949) should be immutable
StakingRewards.stakingToken (FRAXStablecoin.sol#5950) should be immutable
TestSwap.USDT (FRAXStablecoin.sol#4555) should be immutable
TestSwap.USDT_address (FRAXStablecoin.sol#4552) should be immutable
TestSwap.WETH (FRAXStablecoin.sol#4556) should be immutable
TestSwap.WETH_address (FRAXStablecoin.sol#4553) should be immutable
TestSwap.router (FRAXStablecoin.sol#4554) should be immutable
TokenVesting._beneficiary (FRAXStablecoin.sol#3765) should be immutable
TokenVesting._cliff (FRAXStablecoin.sol#3771) should be immutable
TokenVesting._duration (FRAXStablecoin.sol#3773) should be immutable
TokenVesting._owner (FRAXStablecoin.sol#3768) should be immutable
TokenVesting._revocable (FRAXStablecoin.sol#3778) should be immutable
TokenVesting._start (FRAXStablecoin.sol#3772) should be immutable
UniswapV2ERC20.DOMAIN_SEPARATOR (FRAXStablecoin.sol#5445) should be immutable
UniswapV2Pair.factory (FRAXStablecoin.sol#5047) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
FRAXStablecoin.sol analyzed (72 contracts with 84 detectors), 611 result(s) found
```

Solidity Static Analysis

Static code analysis is used to identify many common coding problems before a program is released. It involves examining the code manually or using tools to automate the process. Static code analysis tools can automatically scan the code without executing it.

FRAXStablecoin.sol

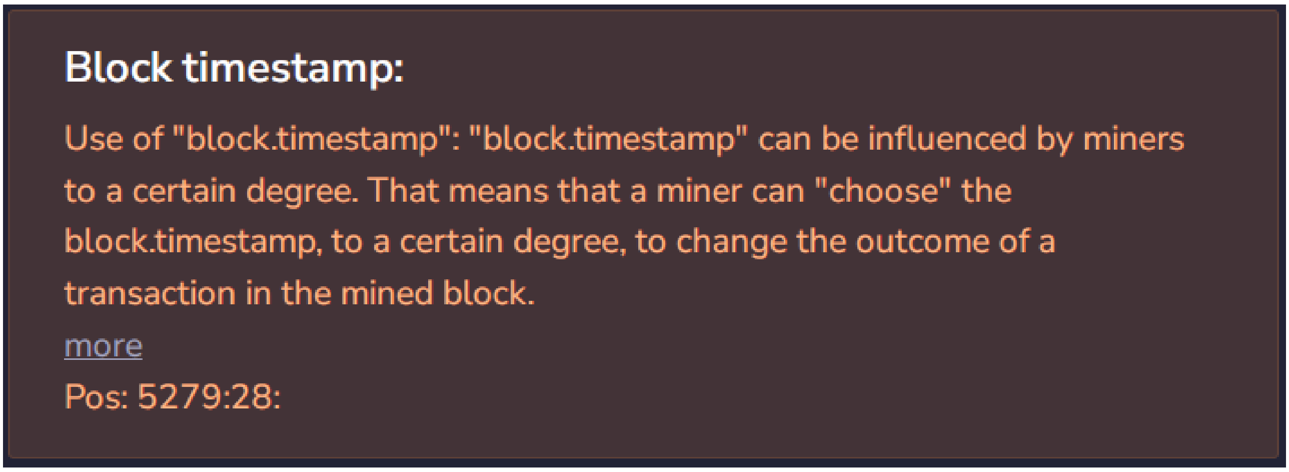


Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

more

Pos: 5551:8:

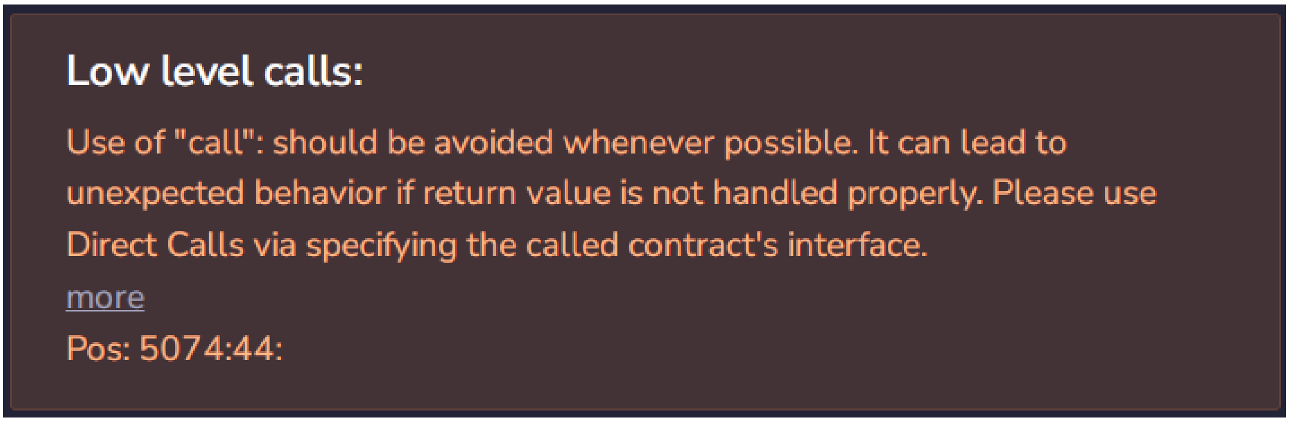


Block timestamp:

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: 5279:28:

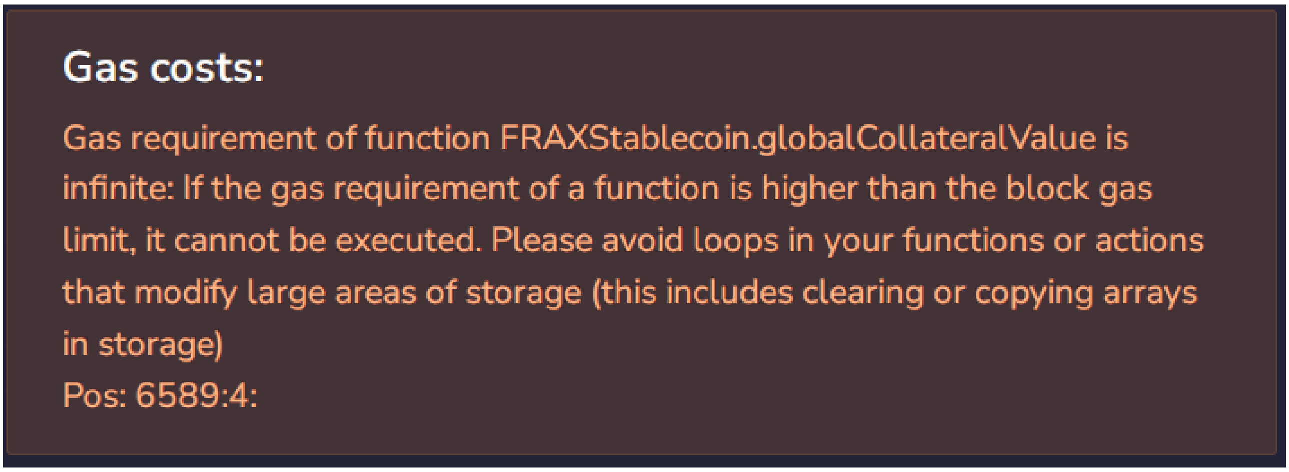


Low level calls:

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

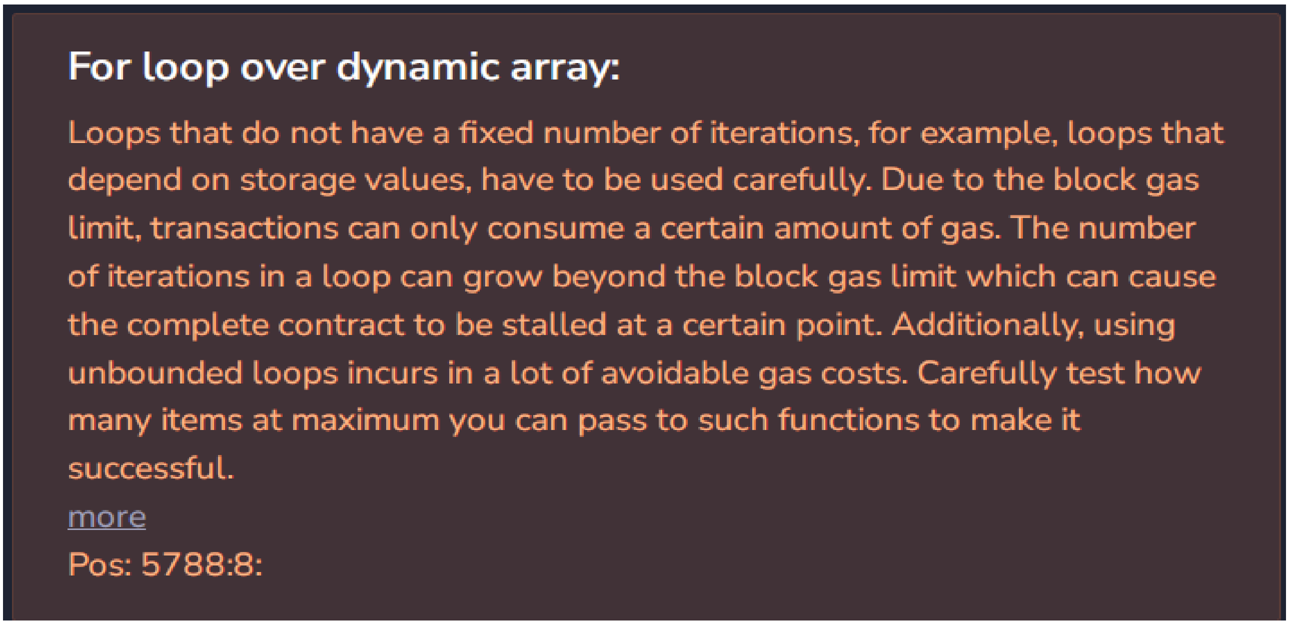
Pos: 5074:44:



Gas costs:

Gas requirement of function FRAXStablecoin.globalCollateralValue 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: 6589:4:

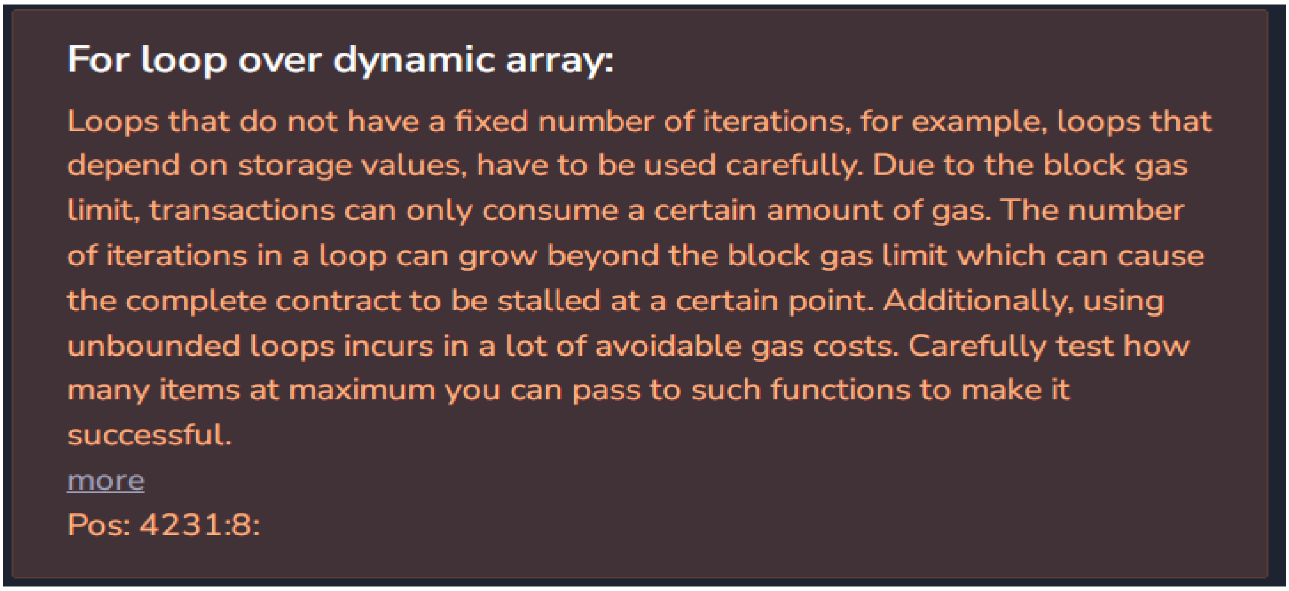


For loop over dynamic array:

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: 5788:8:

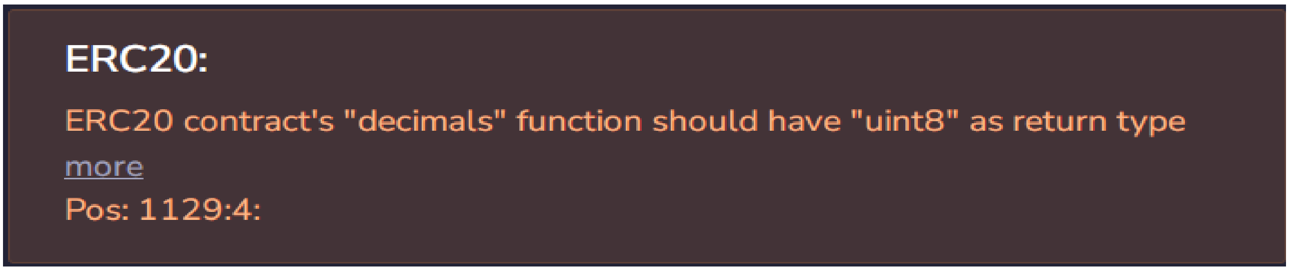


For loop over dynamic array:

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: 4231:8:

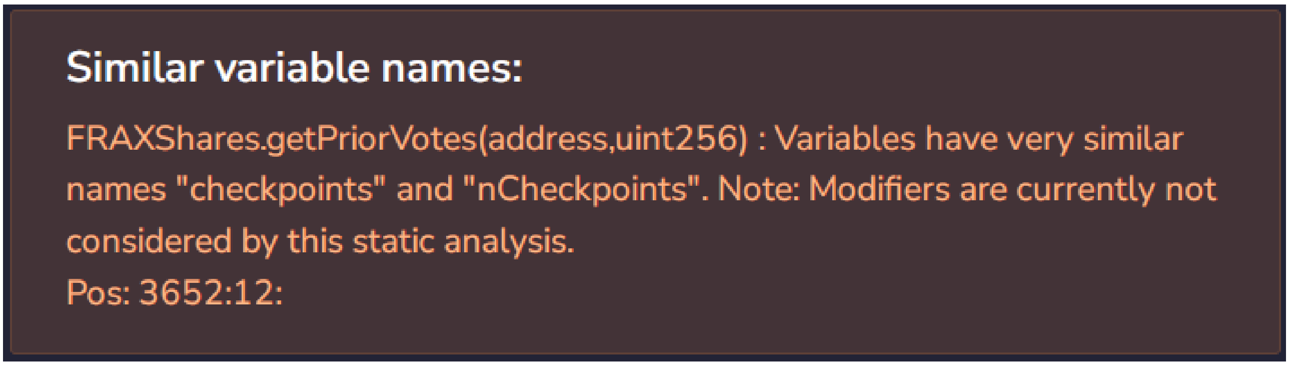


ERC20:

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

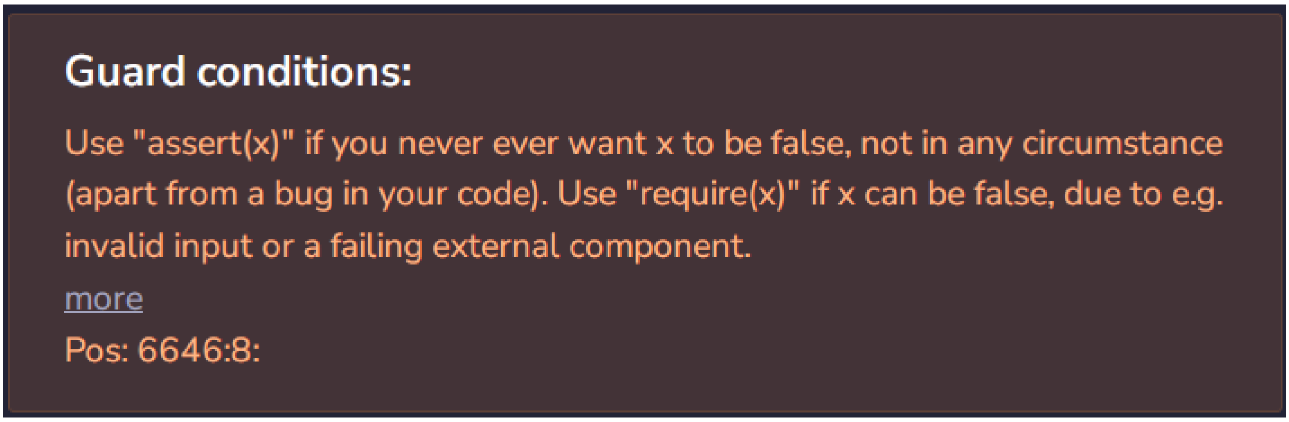
more

Pos: 1129:4:



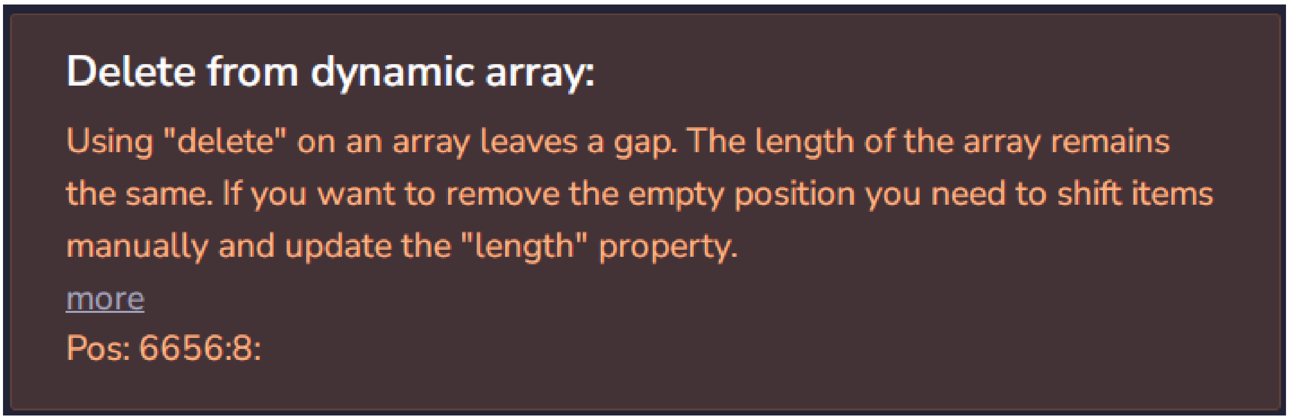
Similar variable names:

FRAXShares.getPriorVotes(address,uint256) : Variables have very similar names "checkpoints" and "nCheckpoints". Note: Modifiers are currently not considered by this static analysis.
Pos: 3652:12:



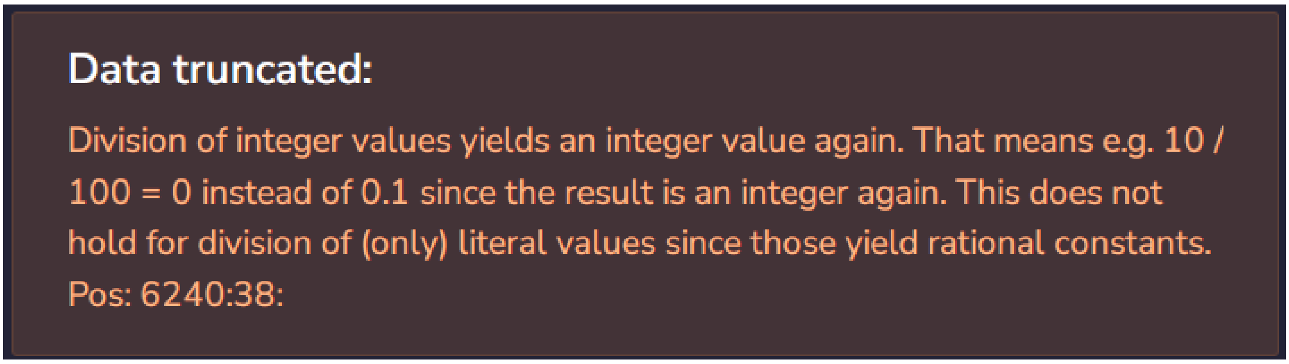
Guard conditions:

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: 6646:8:



Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 6656:8:



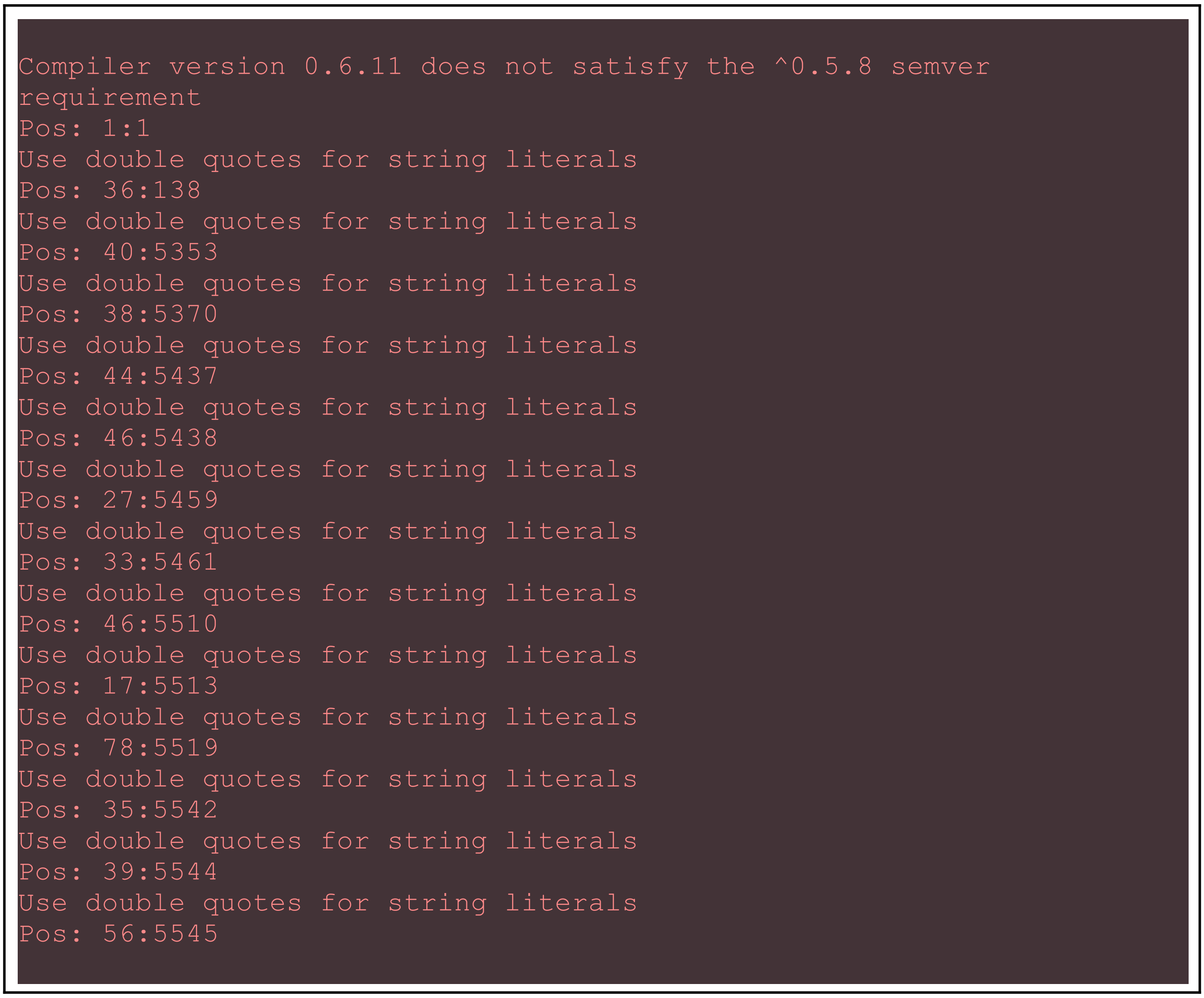
Data truncated:

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: 6240:38:

Solhint Linter

Linters are the utility tools that analyze the given source code and report programming errors, bugs, and stylistic errors. For the Solidity language, there are some linter tools available that a developer can use to improve the quality of their Solidity contracts.

FRAXStablecoin.sol



```
Compiler version 0.6.11 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
Use double quotes for string literals
Pos: 36:138
Use double quotes for string literals
Pos: 40:5353
Use double quotes for string literals
Pos: 38:5370
Use double quotes for string literals
Pos: 44:5437
Use double quotes for string literals
Pos: 46:5438
Use double quotes for string literals
Pos: 27:5459
Use double quotes for string literals
Pos: 33:5461
Use double quotes for string literals
Pos: 46:5510
Use double quotes for string literals
Pos: 17:5513
Use double quotes for string literals
Pos: 78:5519
Use double quotes for string literals
Pos: 35:5542
Use double quotes for string literals
Pos: 39:5544
Use double quotes for string literals
Pos: 56:5545
```

Software analysis result:

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