Frax

Security Assessment

November 6th, 2020

For :

[Travis Moore] @ [Frax]

Disclaimer

CertiK reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

What is a CertiK report?

A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
Representation that a Client of CertiK has indeed completed a round of auditing with the intention to increase the quality of the company/product’s IT infrastructure and or source code.

Overview

Project Summary

Project Name	Frax Protocol
Description	2-token, fully-autonomous protocol which transitions a fully col- lateralized stablecoin (FRAX) to fully algorithmic, moving through a fractional-collateral phase.
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	1. 663f962bd8526c6aadb58e9d56daad907babe813, 2. 3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5, 3. 94bf34cd1157668d03fa076ec53d50f6ce56865b 4. 70f3c859aa82aa95ee163223f2fb3637f9fa97ce

Audit Summary

Delivery Date	Nov. 6, 2020
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	Oct. 11, 2020 - Oct. 30 2020

Vulnerability Summary

Total Issues	39
Total Critical	3
Total Major	8
Total Minor	11
Total Informational	17

Executive Summary

The report represents the results of our engagement with FRAX on their Frax stablecoin protocol. The initial review was conducted for 10 days: Oct. 11, 2020 - Oct. 20 2020 by Adrian Hetman and Alex Papageorgiou.

Initially we found two critical issues, one re-entrancy attack and owner array manipulation that were quickly addressed by the team. Remediations went quickly and next commit hash we were checking solved a lot of previously addressed issues but also we found more issues that were needed to be resolved. Team was quick to engage in the discussion and solving critical and major problems and as last remediations showed, most of the issues were fixed.

Team decided to implement additional logic for slippage protection for all of their minting, redeeming and buyback/recollateralize functions.

Findings

ID	Title	Type	Severity	Resolved
FXS-01	Incorrect version of solidity	Implementation	Minor	✓
FXS-02	Lock solidity version	Implementation	Minor	✓
FXS-03	Example contract from Uniswap used in the project	Implementation	Minor	!
FXS-04	Redundant array of all proposal ids	Implementation	Informational	✓
FXS-05	Solidity changes used from newer versions than pragma entails	Implementation	Minor	✓
FXS-06	Duplication of code in Uniswap implementation.	Implementation	Minor	!
FXS-07	Re-entrancy attack on collateral tokens on Frax Pool	Implementation	Critical	✓
FXS-08	Compilation fails on uncommented code for ChainlinkETHUSDPriceConsumer.sol	Implementation	Major	!
FXS-09	Never initialized governance variable	Implementation	Major	✓
FXS-10	Visibility of variables are not specified.	Implementation	Minor	!
FXS-11	Never used variable	Implementation	Informational	✓
FXS-12	Mark variables as contants	Optimization	Informational	✓
FXS-13	`onlyByOwnerOrGovernance` modifier doesn't check governance address	Implementation	Major	✓
FXS-14	Gas optimization on owner array	Implementation	Informational	✓
FXS-15	Lack of address verification during function call.	Implementation	Major	✓
FXS-16	Owner array manipulation	Implementation	Critical	

				✓
FXS-17	General approach to roles within the system	Implementation	Major	✓
FXS-18	Modifier `onlyByOracle()` allows timelock_address to perform operation.	Implementation	Minor	✓
FXS-19	FXS contract is still using Comp.sol naming.	Implementation	Informational	✓
FXS-20	Require's reason strings naming doesn't reffer to the project	Implementation	Informational	✓
FXS-21	Reason string not present in require	Implementation	Informational	✓
FXS-22	Commented code	Implementation	Informational	✓
FXS-23	Missing natspec comments	Implementation	Informational	!
FXS-24	Usage of literals instead of constant variables	Implementation	Informational	✓
FXS-25	Lack of usage of SafeERC20 from OpenZeppelin	Implementation	Minor	!
FXS-26	Pool's collateral addresses should be defined as constants	Implementation	Informational	!
FXS-27	Owner can change FRAX,FXS and Collateral token address after pool is deployed	Implementation	Major	!
FXS-28	Reduntant initialization.	Implementation	Informational	!
FXS-29	Boolean equality	Implementation	Informational	!
FXS-30	Inefficient greater-than comparison w/ zero	Implementation	Informational	!
FXS-31	Re-entrancy attack in `buyBackFXS()` in FraxPool	Implementation	Critical	✓
FXS-32	Potential for overflow	Implementation	Major	✓

FXS-33	block.number can reach the limit of uint32	Implementation	Minor	
FXS-34	Empty function	Implementation	Minor	
FXS-35	Use of if instead of require	Implementation	Informational	
FXS-36	Require checks could be simplified	Implementation	Informational	
FXS-37	Owner could be set again after team renounces ownership	Implementation	Major	
FXS-38	Oracle address variable not utilized	Implementation	Minor	
FXS-39	Outdated comment	Implementation	Informational	

FXS-01:Incorrect version of solidity

Type	Severity	Location
Implementation	Minor	General

Description:

The linked contracts necessitate a version too recent to be trusted. Consider deploying with 0.6.11. We do not recommend using any latest version for deployment, especially if any changes were made in the optimizer or the language semantic. Version 0.6.12 made changes to the optimizer that's why we do not recommend using this version.

Recommendation:

Deploy with any of the following Solidity versions:

- 0.6.8,
- 0.6.10 - 0.6.11. Use a simple pragma version that allows any of these versions. Consider using the latest version of Solidity for testing.

Alleviations

Alleviations were applied in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5` as advised and the compiler version was locked to `v0.6.11`. Hovewer listed contracts still need to update solidity versions as they are using `pragma ^0.6.0` :

1. Utils/EnumerableSet.sol
2. Frax/Pools/FraxPoolLibrary.sol
3. Utils/SafeMath.sol
4. Governance/AccessControl.sol

FXS-02: Pragma version not locked

Type	Severity	Location
Implementation	Informational	General

Description:

Contract uses `pragma solidity ^0.6.0 <0.7.0;` which is not recommended. Pragmas should be locked to specific compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler, which may have higher risks of undiscovered bugs.

Recommendation:

Avoid a floating pragma version (i.e. `pragma solidity ^0.6.0;` or `version>=0.6.0`) instead specify pragma version without using the caret symbol, i.e., `pragma solidity 0.6.11;`

Alleviations

Alleviations were applied in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5` as advised and the compiler version was locked to `v0.6.11`. Hovewer listed contracts still need to update solidity versions as they are using `pragma ^0.6.0` :

1. Utils/EnumerableSet.sol
2. Frax/Pools/FraxPoolLibrary.sol
3. Utils/SafeMath.sol
4. Governance/AccessControl.sol

FXS-03: Example contract from Uniswap used in the project

Type	Severity	Location
Implementation	Minor	SwapToPrice.sol

Description:

Contract `SwapToPrice.sol` is taken from ExampleSwapToPrice.sol found in Uniswap repository. As noted in their README.md, these implementations should not be assumed to be secure as they are meant to be tests rather than full-blown implementations.

Recommendation:

Avoid usage of example contracts from Uniswap in a production env and move all contracts used for testing purposes to a designated folder, only for test contracts to avoid any confusion.

Alleviations

Alleviations were not applied as advised in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5`. `SwapToPice.sol` is still present in Uniswap folder. The team will be fixing the issues in the own timeframe.

FXS-04: Redundant array of all proposal ids

Type	Severity	Location
Optimization	Informational	Governance.sol L118

Description:

IDs are sequentially increased thus there's no need for an array to hold IDs of proposals

Recommendation:

IDs can easily be generated by knowing the value of `proposalCount` which is a public variable. In that case, we recommend the removal of an array.

Alleviations

Alleviations were applied as advised in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5`. Issue resolved.

FXS-05: Solidity changes used from newer versions than pragma entails

Type	Severity	Location
Implementation	Minor	Timelock.sol L100

Description:

New Call() syntax appeared in solidity 0.6.2 whereas pragma version of solidity is `pragma solidity ^0.6.0 <0.7.0;`

Recommendation:

Use one of the recommended solidity versions from FXS-01.

Alleviations

Alleviations were applied as advised in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5`. Issue resolved.

FXS-06: Duplication of code in Uniswap implementation.

Type	Severity	Location
Implementation	Minor	UniswapV2Pair.sol L17-L22, UniswapV2ERC20.sol L10-L15

Description:

UniswapV2Pair.sol shadows and duplicate UniswapV2ERC20 crucial variables like `totalSupply` or `balanceOf`. `UniswapV2ERC20` should be inherited in `UniswapV2Pair.sol` and depend on its implementation of functions and variable storage.

Recommendation:

We recommend inheriting `UniswapV2ERC20.sol` in `UniswapV2Pair.sol` contract instead of duplicating the code.

Alleviations

Alleviations were not applied as advised in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5`. The team will be fixing the issues in the own timeframe.

FXS-07: Re-entrancy attack on collateral tokens on Frax Pool

Type	Severity	Location
Implementation	Critical	FraxPool.sol L278-L291

Description:

During redeeming of collateral balances in `collectRedemption`, `collateral_token.transfer()` is performed first before marking `redeemCollateralBalances = 0`. Some implementations of ERC20 and ERC777 tokens like `imBTC` can inform the recipient of token transfer with callback call. This can lead to potential re-entrancy if the affected function calls change state variables for balance after transfers are done.

Recommendation:

It is recommended to follow checks-effects-interactions pattern for cases like this. It shields public functions from re-entrancy attacks. It's always a good practice to follow this pattern.

Alleviations

Alleviations were applied as advised in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5`. Issue resolved.

FXS-08: Compilation fails on uncommented code for ChainlinkETHUSDPriceConsumer.sol

Type	Severity	Location
Implementation	Critical	ChainlinkETHUSDPriceConsumer.sol

Description:

Compilation fails in importing `@chainlink/contracts/src/v0.6/interfaces/AggregatorV3Interface.sol";` when code's sections are uncommented as recommended by the the team.

Recommendation:

Import statements should be fixed in the final project repository and contract code for testing should be moved to a designated folder, only for test contracts to avoid any confusion.

Alleviations

Alleviations were partially applied as advised in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5`. Compilation still fails but designed contract for testing was created. The team will be fixing the issues in the own timeframe.

FXS-09: Never initialized governance variable.

Type	Severity	Location
Implementation	Major	Frax.sol L29, Frax.sol L67-L70

Description:

`address governance_address` is never initialized and is used on modifier `onlyByGovernance`. This can lead to functions calls reverting which depends on this modifier.

Recommendation:

Initialize `address governance_address` during contract deployment in the contract's constructor.

Alleviations

Alleviations were applied as advised in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5`. `address governance_address` is replaced by direct usage of `timelock_address`. Issue resolved.

FXS-10: Visibility of variables are not specified.

Type	Severity	Location
Implementation	Minor	Frax.sol L29, Frax.sol L21-L25, Frax.sol L42, FXS.sol L20

Description:

Certain state variables don't have specified visibility. Labeling the visibility explicitly will make it easier to catch incorrect assumptions about who can call the function or access the variable.

Recommendation:

Label the visibility of state variables.

Alleviations

Issue partially resolved. Alleviations were applied as advised in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5`. `address governance_address` is replaced by direct usage of `timelock_address`. `lastRedeemed` mapping in FraxPool.sol L35 doesn't have visibility specified.

FXS-11: Never used variable

Type	Severity	Location
Optimization	informational	Frax.sol L21

Description:

`price_choices` are declared but never used throughout the code. This can lead to wrong assumptions whether the variables are utilized and generate extra gas cost during deployment.

Recommendation:

Remove the variable if it's not needed.

Alleviations

Alleviations were applied as advised in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5`. Issue resolved.

FXS-12: Mark variables as contants

Type	Severity	Location
Optimization	Informational	Frax.sol L27, Frax.sol L37, FXS.sol L17, FXS.sol L16

Description:

Constant state variables should be declared constant to save gas.

Recommendation:

Add the constant attributes to state variables that never change.

Alleviations

Alleviations were applied as advised in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5`. Issue resolved.

FXS-13: `onlyByOwnerOrGovernance` modifier doesn't check governance address

Type	Severity	Location
Informational	Major	FraxPool.sol L52-L63, Frax.sol L72-L83

Description:

`onlyByOwnerOrGovernance` checks only Owners array, does not check Governance address at all. This can lead to potential breaches into the system

Recommendation:

Add logic for checking the governance address to the modifier.

Alleviations

Alleviations were applied as advised in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5`. `timelock_address` is checked in the modifier. Issue resolved.

FXS-14: Gas optimization on owner array

Type	Severity	Location
Optimization	Informational	Frax.sol L28, FraxPool.sol L22

Description:

Operations on arrays are more expensive than mapping when dealing with variables that take the whole storage slot in EVM.

Recommendation:

`owner` array should instead be converted to a mapping to reduce the gas cost involved in the lookup operation.

Alleviations

Alleviations were applied as advised in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5`. Issue resolved.

FXS-15: Lack of address verification during function call.

Type	Severity	Location
Implementation	Major	Frax.sol L212-L229

Description:

Address verification is not present in `addPool` and `removePool` functions. This can lead to duplicates existing within the array. Additionally, the lookup operation where an empty slot is left in the array is very costly in cases many pools are introduced.

Recommendation:

Introduce address verification before function call in form of a require statement or additional modifier.

Alleviations

Alleviations were applied as advised in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5`. Issue resolved.

FXS-16: Owner array manipulation.

Type	Severity	Location
Implementation	Critical	Frax.sol L248-L261, FraxPool L392-L405

Description:

Any of the owners can call `addOwner` or `removeOwner` without any restrictions. This is possible because of the used modifier and its implementation i.e. `onlyByOwnerOrGovernance`. If one of the owners decides to become malicious, it can clear the owner array and become the sole owner of the whole Frax token.

Recommendation:

The form of majority voting mechanism should be introduced in the addition and removal of owners.

Alleviations

Alleviations were applied as advised in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5`. Issue resolved.

FXS-17: General approach to roles within the system.

Type	Severity	Location
Implementation	Major	Frax.sol

Description:

Based on the modifiers and roles used in Frax token, there isn't enough granularity that would help mitigate risks of malicious owner gaining access or owner going rouge and exploiting the system

Recommendation:

We recommend adding more roles and having a more granular approach to roles, dividing parts of the system to be only affected by certain groups. OpenZeppelin implementations of Roles could be used to help to resolve this issue.

Alleviations

The intent is to only have the owners be the project wallet account and the governance contract. Team are never planning on having multiple owners.

At all times, there will ever only be a maximum of 2 addresses that can be owners of any role: the team and timelock. The team will eventually give up their access to certain roles in a staggered fashion and yield full ownership to only the timelock address.

FXS-18: Modifier `onlyByOracle()` allows timelock_address to perform operation.

Type	Severity	Location
Implementation	Minor	FXS.sol L48-L51

Description:

Modifier `onlyByOracle()` checks oracle address as well as timelock_address. Name of the modifier can be misleading for anyone checking the code.

Recommendation:

It's recommended to use more granular approach to roles in the system and have appropriate naming which relfects the access to the function. In this case we recommend changing the name or removing check for `timelock_address`.

Alleviations

Alleviations were applied as advised in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5`. Issue resolved.

FXS-19: FXS contract is still using Comp.sol naming.

Type	Severity	Location
Implementation	Informational	FXS.sol L183-L201, FXS.sol L115, FXS.sol L124

Description:

FXS.sol is based off Comp.sol from Compound. During code rewrite naming from Comp.sol is stil remaining, like function name `_moveDelegates` which is marked as `misnomer`.

Recommendation:

It's recommended to name functions accordingly to its function and what code is doing so it won't be any misunderstanding when studying the code.

Alleviations

Alleviations were applied as advised in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5`. Issue resolved.

FXS-20: Require's reason strings naming doesn't reffer to the project

Type	Severity	Location
Implementation	Informational	FXS.sol L148, FXS.sol L190, FXS.sol L197, FXS.sol L204

Description:

Require's reason string in FXS.sol reffer to the `Comp` instead of the `FXS`.

Recommendation:

Change every `COMP` occurance with `FXS`.

Alleviations

Alleviations were applied as advised in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5`. Issue resolved.

FXS-21: Reason string not present in require

Type	Severity	Location
Implementation	Informational	Frax.sol L168, FraxPool L353,

Description:

Require statements are missing reason strings.

Recommendation:

For the user experience and developers integrating with the protocol, reasons strings in require statements should be present.

Alleviations

Alleviations were applied as advised in commit `70f3c859aa82aa95ee163223f2fb3637f9fa97ce`. Issue resolved.

FXS-22: Commented code

Type	Severity	Location
Implementation	Informational	FraxPool.sol L136

Description:

There is a commented code which is not used.

Recommendation:

Commented code should be removed from the contract.

Alleviations

Alleviations were applied as advised in commit `70f3c859aa82aa95ee163223f2fb3637f9fa97ce`. Issue resolved.

FXS-23: Missing natspec comments

Type	Severity	Location
Implementation	Informational	FraxPools, Frax.sol, FXS.sol, Uniswap Contracts, Oracle Contracts, Governance.sol, Timelock.sol

Description:

Contract code is missing natspec comments, which helps understand the code and all the functions' parameters.

Recommendation:

Please follow these style guides for adding natspec comments. https://solidity.readthedocs.io/en/v0.6.11/style-guide.html?highlight=natspec#natspec

Alleviations

Alleviations were not applied as advised in commit `3b74b6bdc31c3f97f3f62e2462bdecfd84418dc5`. The team will be fixing the issues in the own timeframe.

FXS-24: Usage of literals instead of constant variables

Type	Severity	Location
Implementation	Informational	FraxPool L153, FraxPool L131

Description:

Literals with many digits are difficult to read and are hard to maintain in terms of code changes.

Recommendation:

It is recommended to convert literals to the constant values and use them instead. This way, code will look cleaner, and it will be easier to maintain. Every occurance of `1e6` or `1000000` should be instead converted to one constant value with corresponsing name to its representation.

Alleviations

Alleviations were applied as advised in commit `70f3c859aa82aa95ee163223f2fb3637f9fa97ce`. Issue resolved.

FXS-25: Lack of usage of SafeERC20 from OpenZeppelin

Type	Severity	Location
Implementation	Minor	FraxPool L164, FraxPool L206, FraxPool L223, FraxPool L244, FraxPool L249, FraxPool L266, FraxPool L281, FraxPool L287, FraxPool L311, FraxPool L348, FraxPool L261-L262, FraxPool L346, FraxPool L283

Description:

While the ERC-20 implementation does necessitate that the `transferFrom()` / `transfer()` function returns a `bool` variable yielding `true`, many token implementations do not return anything i.e. Tether (USDT) leading to unexpected halts in code execution.

Recommendation:

We advise that the `SafeERC20.sol` library is utilized by OpenZeppelin to ensure that the `transferFrom()` / `transfer()` function is safely invoked in all circumstances.

Alleviations

Alleviations were not applied as advised in commit `70f3c859aa82aa95ee163223f2fb3637f9fa97ce`. The team will be fixing the issues in the own timeframe.

FXS-26: Pool's collateral addresses should be defined as constants

Type	Severity	Location
Implementation	informational	Pool_USDC L8, Pool_USDT L8, Pool_ySDC L8

Description:

Pool contracts for specific stablecoins don't have constant variables with addresses of said stablecoin.

Recommendation:

It is recommended to use constant variables inside `Pool_X` contracts that specify said stablecoin pool. This shows the community collateral token address can't be tempered with.

Alleviations

Alleviations were not applied as advised in commit `70f3c859aa82aa95ee163223f2fb3637f9fa97ce`. The team will be fixing the issues in the own timeframe.

FXS-27: Owner can change FRAX, FXS, Timelock, Oracle, Owner and Collateral token address after pool is deployed

Type	Severity	Location
Implementation	Major	FraxPool L376-L389, FraxPool L376-L389

Description:

Functions `setCollateralAdd`, `setFRAXAddress`, `setFXSAddress`, `setOracle`, `setTimelock` and `setOwner` allows change of important variables which should be defined once upon pool creation. Malicious owner could use these functions to benefit himself.

Recommendation:

Remove these functions from the code or add governance vote before being able to call them.

Alleviations

Alleviations were partially applied as advised in commit `70f3c859aa82aa95ee163223f2fb3637f9fa97ce`. Based on commit (2.), issue was updated with recommendation regarding `setOracle`, `setTimelock` and `setOwner` functions.

FXS-28: Reduntant initialization.

Type	Severity	Location
Implementation	Informational	FraxPool L54-L57, FraxPool L42, FraxPool L45, Frax L54

Description:

redundant initialization of storage variables as it will default to given values during compilation as default.

Recommendation:

Do not initialize storage variables with default values.

Alleviations

Alleviations were not applied as advised in commit `70f3c859aa82aa95ee163223f2fb3637f9fa97ce`. The team will be fixing the issues in the own timeframe.

FXS-29: Boolean equality

Type	Severity	Location
Implementation	Informational	FraxPool L131, FraxPool L296, FraxPool L299, FraxPool L308, FraxPool L370, FraxPool L67, FraxPool L72, Frax L163, Frax L209, Frax L216, Frax L64, Frax L74-L78, FXS.sol 44, Governance.sol L279

Description:

Boolean variables can be used directly and do not need to be compared to true or false.

Recommendation:

Remove comparision with true or false for boolean variables.

Alleviations

Alleviations were not applied as advised in commit `70f3c859aa82aa95ee163223f2fb3637f9fa97ce`. The team will be fixing the issues in the own timeframe.

FXS-30: Inefficient greater-than comparison w/ zero

Type	Severity	Location
Optimization	Informational	FraxPool L178, FraxPool L228, FraxPool L280, FraxPool L288, FXS.sol L143, FXS.sol L192, FXS.sol L195, FXS.sol L202, FXS.sol L212,Governance.sol L241, StakingRewards.sol L185, StakingRewards.sol L201-L202, StakingRewards.sol L231, StakingRewards.sol L260, StakingRewards.sol L

Description:

Within Solidity, unsigned integers are restricted to the non-negative range. As such, greater-than comparisons with the literal `0` are inefficient gas-wise.

Recommendation:

Consider converting the linked comparisons to inequality ones in order to optimize their gas cost.

Alleviations

Alleviations were not applied as advised in commit `70f3c859aa82aa95ee163223f2fb3637f9fa97ce`. The team will be fixing the issues in the own timeframe.

FXS-31: Re-entrancy attack in `buyBackFXS()` in FraxPool

Type	Severity	Location
Implementation	critical	FraxPool L307-L324

Description:

Re-entrancy here would be possible whereby a user gets the collateral first and uses it to buy the necessary FXS to burn for on L323 i.e. `FXS.burnFrom(msg.sender, FXS_amount);`. Some implementations of ERC20 and ERC777 tokens like `imBTC` can inform the recipient of token transfer with callback call.

Recommendation:

Overall tokens should first be burned. Follow checks-effects-interactions pattern

Alleviations

Alleviations were applied as advised in commit `94bf34cd1157668d03fa076ec53d50f6ce56865b`. Tokens are burned first then transfer happens. Issue resolved.

FXS-32: Potential for overflow

Type	Severity	Location
Implementation	Major	FXS.sol L97, FXS.sol L103

Description:

Operations with `+` and `-` should be using safeMath as this could lead to overflow and underflow.

Recommendation:

It's always recommended to use SafeMath's functions for any arithmetic operations.

Alleviations

Alleviations were applied as advised in commit `94bf34cd1157668d03fa076ec53d50f6ce56865b`. SafeMath was used on arithmetic operations. Issue resolved.

FXS-33: block.number can reach the limit of uint32

Type	Severity	Location
Implementation	Minor	FXS.sol L210

Description:

Usage of safe32 on block.number is dangerous as block.number can reach the limit of uint32 in the future i.e. 4 billion blocks depending on tech advancement, EOS is already at 100+ million blocks in only 2 years.

Recommendation:

Change it to uint256.

Alleviations

Alleviations were not applied as advised in commit `70f3c859aa82aa95ee163223f2fb3637f9fa97ce`. The team will be fixing the issues in the own timeframe.

FXS-34: Empty function.

Type	Severity	Location
Implementation	Minor	StakingRewards.sol L329-L351

Description:

Implementation of the function `notifyRewardAmount()` is commented out.

Recommendation:

Remove function or uncomment the code.

Alleviations

Alleviations were not applied as advised in commit `70f3c859aa82aa95ee163223f2fb3637f9fa97ce`. The team will be fixing the issues in the own timeframe.

FXS-35: Use of if instead of require

Type	Severity	Location
Implementation	Informational	StakingRewards.sol L299-L301

Description:

```
if (block.timestamp > periodFinish) {
            retroCatchUp();
        }
```

Instead of if, require could be use.

Recommendation:

Change if to require.

Alleviations

Alleviations were not applied as advised in commit `70f3c859aa82aa95ee163223f2fb3637f9fa97ce`. The team will be fixing the issues in the own timeframe.

FXS-36: Require checks could be simplified

Type	Severity	Location
Implementation	Informational	StakingRewards.sol L202

Description:

`secs` can never be negative and `locked_stake_min_time` is guaranteed to be gte to 1.

Recommendation:

Requirement statements could be simplified to take into account these properties.

Alleviations

Alleviations were not applied as advised in commit `70f3c859aa82aa95ee163223f2fb3637f9fa97ce`. The team will be fixing the issues in the own timeframe.

FXS-37: Owner could be set again after team renounces ownership

Type	Severity	Location
Implementation	Major	FraxPool L401-L403

Description:

Function `setOwner` allows change of owner of the contract. As team wants to renounce of the ownership of the contract after few weeks and allow of only governance to make changes to the contract, this function would still allow to set new owner even if team would renounce ownership of the contract.

Recommendation:

We recommend changing logic of this function from setting a new owner to renouncing ownership all together. Similar function can be found in `Ownable.sol` from OZ.

```
function renounceOwnership() public virtual onlyOwner {
        emit OwnershipTransferred(_owner, address(0));
        _owner = address(0);
    }
```

This way, when team wants to renounce ownership, no new owner could be set by governance.

Alleviations

Alleviations were not applied as advised in commit `70f3c859aa82aa95ee163223f2fb3637f9fa97ce`. The team will be fixing the issues in the own timeframe.

FXS-38: Oracle address variable not utilized

Type	Severity	Location
Implementation	Minor	FraxPool L23, FXS L25

Description:

`oracle_address` variable in FraxPool.sol and in FXS.sol are not utlized in the contracts. They are only set in the constructor and there isn't any usage for it in the code.

Recommendation:

If variable is not needed we recommend to delete it.

Alleviations

Alleviations were partially applied as advised in commit `70f3c859aa82aa95ee163223f2fb3637f9fa97ce`. The team will be fixing the issues in the own timeframe for FXS.sol

FXS-39: Outdated comment

Type	Severity	Location
Implementation	Informational	Frax L35

Description:

Comments states `1M` of token while is `genesis_supply = 2000000e18;`

Recommendation:

Change the comment to match the variable or variable to match the comment.

Alleviations

The team will be fixing the issues in the own timeframe.

Icons explanation

 : Issue resolved

 : Issue not resolved. The team will be fixing the issues in the own timeframe.

 : Issue partially resolved. Not all instances of an issue was resolved.