SMART CONTRACT CODE SECURITY ANALYSIS REPORT

Project : Fortress Lending Platform

Customer : Jetfuel Finance

Date : 12/04/2021

Table of Content

Copyright © 2021 Hash0X (audit@hash0x.com)

Disclaimer

Hash0X 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 Hash0X to perform a security review.

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The content of this audit report is provided “as is”, without representations and warranties of any kind, and Hash0X disclaims any liability for damage arising out of, or in connection with, this audit report. Copyright of this report remains with Hash0X.

Purpose of the report

The Audits and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of Solidity code and only the Solidity code we note as being within the scope of our review within this report. The Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the Solidity programming language that could present security risks. Cryptographic tokens and smart contracts are emergent technologies and carry with them high levels of technical risk and uncertainty.

The Audits are not an endorsement or indictment of any particular project or team, and the Audits do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset.

Introduction

We first thank Fortress Team for giving us the opportunity to audit their smart contract. This document outlines our methodology, audit details, and results. Fortress Team asked us to review their Fortress Protocol smart contracts.

Hash0X reviewed the system from a technical perspective looking for bugs, issues and vulnerabilities in their code base. This audit report is valid for the smart contract at the mentioned commit hashes only. This audit report is not valid for any other versions of the smart contract files.

Project files

File Name	MD5 Hash
FTS.sol	16184E612400DCE0013F54FB60212FF2
FAI.sol	D09CB24C6EA078EEC2F38348244E215C
Unitroller.sol	2CA395D65CCA9872B141A39761850117
Comptroller.sol	B9215D650D78D056DBF26DFB949FF9DF
FAIUnitroller.sol	D363232ED72C9F0B78DE8AB0140EF5D3
FAIController.sol	28668F1BCD8DD4BFA514789790842396
SFTVaultProxy.sol	5B2F3BA1C4777003C7BF7AE3D1914043
SFTVault.sol	538ABD88AFDC40BC1D1487216BCB5F58
FortressLens.sol	CDEA199BB76B31007252700B9469371C
WhitePaperInterestRateModel.sol	111F06FACC068AC86133F754F4396F40
FortressPriceOracle.sol	4E39ACF0B27511860B168F6C76C85B09
FBep20Delegate.sol	F48021DCC4AF0D5AFA2B9712C61484E7
FBep20Delegator.sol	5993302613299E41D2043DF3459D858C
Timelock.sol	8025863D3B4036F7FFC40E53CAD717AB
GovernorAlpha.sol	F6CFEA696869B67C6414B4616EA0A6F6

Copyright © 2021 Hash0X (audit@hash0x.com)

Audit Summary

Several issues were found in the first audit which were fixed by the Fortress Team and no issues were found in the revised audit of the patched version.

High Severity Issues Found	0
Moderate Severity Issues Found	0
Low Severity Issues	1
Informational Observations	3

The smart contract is considered to **pass** the audit, as of the audit date, if no high severity or moderate severity issues are found.

Copyright © 2021 Hash0X (audit@hash0x.com)

Overview

The project has 15 core Solidity files for the Fortress Protocol Smart Contract, the Fortress Protocol file that contains imports from some OpenZeppelin smart contract libraries. Code review of OpenZeppelin libraries or servers / backend system is outside the scope of this audit report. We manually reviewed each line of code in the smart contract within the scope.

Methodology

Hash0X manually reviewed the smart contract line-by-line, keeping in mind industry best practices and known attacks, looking for any potential issues and vulnerabilities, and areas where improvements are possible.

We also used automated tools like slither / surya for analysis and reviewing the smart contract. These tools often give false-positives, and any issues reported by them but not included in the issue list can be considered not valid.

Classification / Issue Types Definition:

1. **High Severity**: which presents a significant security vulnerability or failure of the contract across a range of scenarios, or which may result in loss of funds.
2. **Moderate Severity**: which affects the desired outcome of the contract execution or introduces a weakness that can be exploited. It may not result in loss of funds but breaks the functionality or produces unexpected behavior.
3. **Low Severity**: which does not have a material impact on the contract execution and is likely to be subjective.

As mentioned above, the smart contract is considered to pass the audit, as of the audit date, if no high severity or moderate severity issues are found.

Copyright © 2021 Hash0X (audit@hash0x.com)

Attacks & Issues considered while auditing

In order to check for the security of the contract, we reviewed each line of code in the smart contract considering several known Smart Contract Attacks & known issues

- **Potential Issue: Overflows and underflows**

  An overflow happens when the limit of the type variable uint256 , 2 ** 256, is exceeded. What happens is that the value resets to zero instead of incrementing more. For instance, if we want to assign a value to a uint bigger than 2 ** 256 it will simple go to 0—this is dangerous. On the other hand, an underflow happens when you try to subtract 0 minus a number bigger than 0. For example, if you subtract 0 - 1 the result will be = 2 ** 256 instead of -1. This is quite dangerous.

  Finding: This contract DOES check for overflows and underflows using SafeMath libraries.

- **Reentrancy Attack**

  One of the major dangers of calling external contracts is that they can take over the control flow, and make changes to your data that the calling function wasn't expecting. This class of bug can take many forms, and both of the major bugs that led to the DAO's collapse were bugs of this sort.

  Finding: This smart contract uses Check-effect pattern to protect against this attack.

- **Replay attack**

  The replay attack consists of making a transaction on one blockchain like the original Ethereum’s blockchain and then repeating it on another blockchain like the Ethereum’s classic blockchain. The ether is transferred like a normal transaction from a blockchain to another. Though it's no longer a problem because since the version 1.5.3 of Geth and 1.4.4 of Parity both implement the attack protection EIP 155 by Vitalik Buterin. So, the people that will use the contract depend on their own ability to be updated with those programs to keep themselves secure. Since this full system is a cross-chain bridge between Binance Smart Chain and Ethereum Blockchains – it is recommended to not

let users enter arbitrary chain IDs in the transfer and receipt requests which may result in a potential replay attack in the future.

- **Short address attack**

  This attack affects ERC20 tokens, was discovered by the Golem team and consists of the following: A user creates an Ethereum wallet with a trailing 0, which is not hard because it's only a digit. For instance: 0xiofa8d97756as7df5sd8f75g8675ds8gsdg0 Then he buys tokens by removing the last zero: Buy 1000 tokens from account 0xiofa8d97756as7df5sd8f75g8675ds8gsdg. If the contract has enough amount of tokens and the buy function doesn't check the length of the address of the sender, the Ethereum's virtual machine will just add zeroes to the transaction until the address is complete.

  Finding: This issue is not applicable to Fortress smart contracts.

- **Approval Double-spend**

  ERC20 Standard allows users to approve other users to manage their tokens, or spend tokens from their account till a certain amount, by setting the user's allowance with the standard `approve` function, then the allowed user may use `transferFrom` to spend the allowed tokens. Hypothetically, given a situation where Alice approves Bob to spend 100 Tokens from her account, and if Alice needs to adjust the allowance to allow Bob to spend 20 more tokens, normally – she'd check Bob's allowance (100 currently) and start a new `approve` transaction allowing Bob to spend a total of 120 Tokens instead of 100 Tokens.

  Finding: Likely impact of this bug is low for most situations. For more, see this discussion on GitHub: https://github.com/ethereum/EIPs/issues/20#issuecomment263524729

Issues Found

High Severity Issues

No moderate severity issues were found in the smart contract.

Moderate Severity Issues

No moderate severity issues were found in the smart contract.

Low Severity Issues

(1) Possibility of infinite loop:

```
function execute(uint proposalId) public payable {
    require(state(proposalId) == ProposalState.Queued, "GovernorAlpha::execute: proposal can only be executed if it is queued");
    Proposal storage proposal = proposals[proposalId];
    proposal.executed = true;
    for (uint i = 0; i < proposal.targets.length; i++) {
        timelock.executeTransaction.value(proposal.values[i])(proposal.targets[i], proposal.values[i], proposal.signatures[i], p
proposal.eta);
    }
    emit ProposalExecuted(proposalId);
}
```

In execute function in GovernorAlpha.sol contract as well as many other places, there are loops which are not limited. If they are used with moderation, then its fine. Otherwise, it might hit the block gas limit.

Fix: Fortress team will make sure to use these functions with limited loop iterations.

Informational Observations

(1) Use latest solidity version: It is recommended to use latest solidity version as they fix many compiler levels bugs from the old versions.

(2) Use visibility External over public: If any function is not being called internally, then it is better to specify its visibility as external. It saves some gas as well. https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices

Audit Conclusion

The fortress protocol smart contract codes were written very systematic, that we did not find any major issues in it. Hence, this code is ready for the production.

Due to the nature of the smart contract protocol, there are unlimited use case scenarios, thus it is not possible to give guarantee about the future outcomes. This audit is based on manual code analysis as well as used latest static tools.

This audit report presents all the findings based on standard audit procedure, which includes manual analysis as well as automated software tools. Smart Contract's high-level description of functions was presented in Appendix section of the report.

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

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

Appendix

Smart Contract Functional Summary

FTS.sol

Sl.	Function	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	allowance	read	Passed	No Issue
3	approve	write	Passed	No Issue
4	balanceOf	read	Passed	No Issue
5	transfer	write	Passed	No Issue
6	transferFrom	write	Passed	No Issue
7	delegate	write	Passed	No Issue
8	delegateBySig	write	Passed	No Issue
9	getCurrentVotes	read	Passed	No Issue
10	getPriorVotes	read	Infinite loop possibility	Votes must not be excessive
11	_delegate	internal	Passed	No Issue
12	_transferTokens	internal	Passed	No Issue
13	_moveDelegates	internal	Passed	No Issue
14	_writeCheckpoint	internal	Passed	No Issue
15	safe32	read	Passed	No Issue
16	safe96	read	Passed	No Issue
17	add96	read	Passed	No Issue
18	sub96	read	Passed	No Issue
19	getChainId	read	Passed	No Issue

FAI.sol

Sl.	Function	Type	Observation	Conclusion
1	rely	write	Passed	No Issue
2	deny	write	Passed	No Issue
3	add	read	Passed	No Issue
4	sub	read	Passed	No Issue
5	constructor	write	Passed	No Issue
6	transfer	write	Passed	No Issue
7	transferFrom	write	Passed	No Issue
8	mint	write	No max minting	Unitroller regulates minting
9	burn	write	Passed	No Issue
10	approve	write	Passed	No Issue
11	push	write	Passed	No Issue
12	pull	write	Passed	No Issue
13	move	write	Passed	No Issue
14	permit	write	Passed	No Issue

Unitroller.sol

Sl.	Function	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_setPendingImplementation	write	Passed	No Issue
3	_acceptImplementation	write	Passed	No Issue
4	_setPendingAdmin	write	Passed	No Issue
5	_acceptAdmin	write	Passed	No Issue
6	fallback	write	Delegates to implementation	No Issue

Copyright © 2021 Hash0X (audit@hash0x.com)

Comptroller.sol

Sl	Function	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getAssetsIn	read	Passed	No Issue
3	checkMembership	read	Passed	No Issue
4	enterMarkets	write	Infinite loop possibility	Keep fTokens limited
5	addToMarketInternal	internal	Passed	No Issue
6	exitMarket	write	Passed	No Issue
7	mintAllowed	write	Passed	No Issue
8	mintVerify	write	Passed	No Issue
9	redeemAllowed	write	Passed	No Issue
10	redeemAllowedInternal	internal	Passed	No Issue
11	redeemVerify	write	Passed	No Issue
12	borrowAllowed	write	Passed	No Issue
13	borrowVerify	write	Passed	No Issue
14	repayBorrowAllowed	write	Passed	No Issue
15	repayBorrowVerify	write	Passed	No Issue
16	liquidateBorrowAllowed	write	Passed	No Issue
17	liquidateBorrowVerify	write	Passed	No Issue
18	seizeAllowed	write	Passed	No Issue
19	seizeVerify	write	Passed	No Issue
20	transferAllowed	write	Passed	No Issue
21	transferVerify	write	Passed	No Issue
22	getAccountLiquidity	read	Passed	No Issue
23	getHypotheticalAccountLiquidity	read	Passed	No Issue
24	getHypotheticalAccountLiquidityInternal	internal	Infinite loop possibility	Keep assets limited
25	liquidateCalculateSeizeTokens	read	Passed	No Issue
26	_setPriceOracle	write	Passed	No Issue
27	_setCloseFactor	write	Passed	No Issue
28	_setCollateralFactor	write	Passed	No Issue
29	_setMaxAssets	write	Passed	No Issue
30	_setLiquidationIncentive	write	Passed	No Issue
31	_supportMarket	write	Passed	No Issue
32	_addMarketInternal	internal	Passed	No Issue
33	_setProtocolPaused	write	Passed	No Issue
34	_setFAIController	write	Passed	No Issue
35	_setFAIMintRate	write	Passed	No Issue

36	_setTreasuryData	write	Passed	No Issue
37	_become	write	Passed	No Issue
38	refreshFortressSpeeds	write	Passed	No Issue
39	refreshFortressSpeedsInternal	internal	Infinite loop possibility	Markets must be limited
40	updateFortressSupplyIndex	internal	Passed	No Issue
41	updateFortressBorrowIndex	internal	Passed	No Issue
42	distributeSupplierFortress	internal	Passed	No Issue
43	distributeBorrowerFortress	internal	Passed	No Issue
44	distributeFAIMinterFortress	write	Passed	No Issue
45	transferFTS	internal	Passed	No Issue
46	claimFortress	write	Infinite loop possibility	Array length must be limited
47	_setFortressRate	write	Passed	No Issue
48	_setFortressFAIRate	write	Passed	No Issue
49	_setFortressFAIVaultRate	write	Passed	No Issue
50	_setFAIVaultInfo	write	Passed	No Issue
51	_addFortressMarkets	write	Infinite loop possibility	Array length must be limited
52	_addFortressMarketInternal	internal	Passed	No Issue
53	_dropFortressMarket	write	Passed	No Issue
54	getAllMarkets	read	Passed	No Issue
55	getBlockNumber	read	Passed	No Issue
56	getFTSAddress	read	hard coded address	Keep it in a variable
57	setMintedFAIOf	write	Passed	No Issue
58	releaseToVault	write	Passed	No Issue

FAIUnitroller.sol

Sl.	Function	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_setPendingImplementation	write	Passed	No Issue
3	_acceptImplementation	write	Passed	No Issue
4	_setPendingAdmin	write	Passed	No Issue
5	_acceptAdmin	write	Passed	No Issue
6	fallback	write	Delegates to implementation	No Issue

FAIController.sol

Sl.	Function	Type	Observation	Conclusion
1	mintFAI	write	Passed	No Issue
2	repayFAI	write	Passed	No Issue
3	_initializeFortressFAIState	write	Passed	No Issue
4	updateFortressFAIMintIndex	write	Passed	No Issue
5	calcDistributeFAIMinterFortress	write	Passed	No Issue
6	_setComptroller	write	Passed	No Issue
7	_become	write	Passed	No Issue
8	getMintableFAI	write	Infinite loop possibility	Keep array length limited
9	getBlockNumber	read	Passed	No Issue
10	getFAIAddress	read	hard coded address	Keep it in a variable

SFTVaultProxy.sol

Sl.	Function	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_setPendingImplementation	write	Passed	No Issue
3	_acceptImplementation	write	Passed	No Issue
4	_setPendingAdmin	write	Passed	No Issue
5	_acceptAdmin	write	Passed	No Issue
6	fallback function	write	Delegates to implementation	No Issue

Copyright © 2021 Hash0X (audit@hash0x.com)

SFTVault.sol

Sl.	Function	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	deposit	write	Passed	No Issue
3	withdraw	write	Passed	No Issue
4	claim	write	Passed	No Issue
5	_withdraw	internal	Passed	No Issue
6	pendingFTS	read	Passed	No Issue
7	updateAndPayOutPending	internal	Passed	No Issue
8	safeFTSTransfer	internal	Passed	No Issue
9	updatePendingRewards	write	Passed	No Issue
10	updateVault	internal	Passed	No Issue
11	getAdmin	read	Passed	No Issue
12	burnAdmin	write	Passed	No Issue
13	setNewAdmin	write	Passed	No Issue
14	_become	write	Passed	No Issue
15	setFortressInfo	write	Passed	No Issue

FortressLens.sol

Sl.	Function	Type	Observation	Conclusion
1	fTokenMetadata	write	Passed	No Issue
2	fTokenMetadataAll	write	Infinite loop possibility	Keep array length limited
3	fTokenBalances	write	Passed	No Issue
4	fTokenBalancesAll	write	Infinite loop possibility	Keep array length limited
5	fTokenUnderlyingPrice	read	Passed	No Issue
6	fTokenUnderlyingPriceAll	read	Infinite loop possibility	Keep array length limited
7	getAccountLimits	read	Passed	No Issue
8	getGovReceipts	read	Infinite loop possibility	Keep array length limited
9	setProposal	internal	Passed	No Issue
10	getGovProposals	read	Infinite loop possibility	Keep array length limited
11	getFTSBalanceMetadata	read	Passed	No Issue
12	getFTSBalanceMetadataExt	write	Passed	No Issue
13	getFortressVotes	read	Passed	No Issue

WhitePaperInterestRateModel.sol

Sl.	Function	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	utilizationRate	read	Passed	No Issue
3	getBorrowRate	read	Passed	No Issue
4	getSupplyRate	read	Passed	No Issue

FortressPriceOracle.sol

Sl.	Function	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getUnderlyingPrice	read	Passed	No Issue
3	setUnderlyingPrice	write	Passed	No Issue
4	setDirectPrice	write	Passed	No Issue
5	assetPrices	read	Passed	No Issue
6	compareStrings	read	Passed	No Issue
7	setAdmin	write	Passed	No Issue

FBep20Delegator.sol

Sl.	Function	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	_resignImplementation	write	Passed	No Issue
3	_setImplementation	write	Passed	No Issue
4	mint	write	Passed	No Issue
5	redeem	write	Passed	No Issue
6	redeemUnderlying	write	Passed	No Issue
7	borrow	write	Passed	No Issue
8	repayBorrow	write	Passed	No Issue
9	repayBorrowBehalf	write	Passed	No Issue
10	liquidateBorrow	write	Passed	No Issue
11	transfer	write	Passed	No Issue
12	transferFrom	write	Passed	No Issue
13	approve	write	Passed	No Issue
14	allowance	read	Passed	No Issue
15	balanceOf	read	Passed	No Issue
16	balanceOfUnderlying	write	Passed	No Issue
17	getAccountSnapshot	read	Passed	No Issue
18	borrowRatePerBlock	read	Passed	No Issue

Copyright © 2021 Hash0X (audit@hash0x.com)

19	supplyRatePerBlock	read	Passed	No Issue
20	totalBorrowsCurrent	write	Passed	No Issue
21	borrowBalanceCurrent	write	Passed	No Issue
22	borrowBalanceStored	read	Passed	No Issue
23	exchangeRateCurrent	write	Passed	No Issue
24	exchangeRateStored	read	Passed	No Issue
25	getCash	read	Passed	No Issue
26	accrueInterest	write	Passed	No Issue
27	seize	write	Passed	No Issue
28	_setPendingAdmin	write	Passed	No Issue
29	_setComptroller	write	Passed	No Issue
30	_setReserveFactor	write	Passed	No Issue
31	_acceptAdmin	write	Passed	No Issue
32	_addReserves	write	Passed	No Issue
33	_reduceReserves	write	Passed	No Issue
34	_transferReserves	write	Passed	No Issue
35	_setInterestRateModel	write	Passed	No Issue
36	delegateTo	internal	Passed	No Issue
37	delegateToImplementation	write	Passed	No Issue
38	delegateToViewImplementation	read	Passed	No Issue
39	delegateToViewAndReturn	read	Passed	No Issue
40	delegateAndReturn	write	Passed	No Issue
41	fallback function	write	Passed	No Issue

Timelock.sol

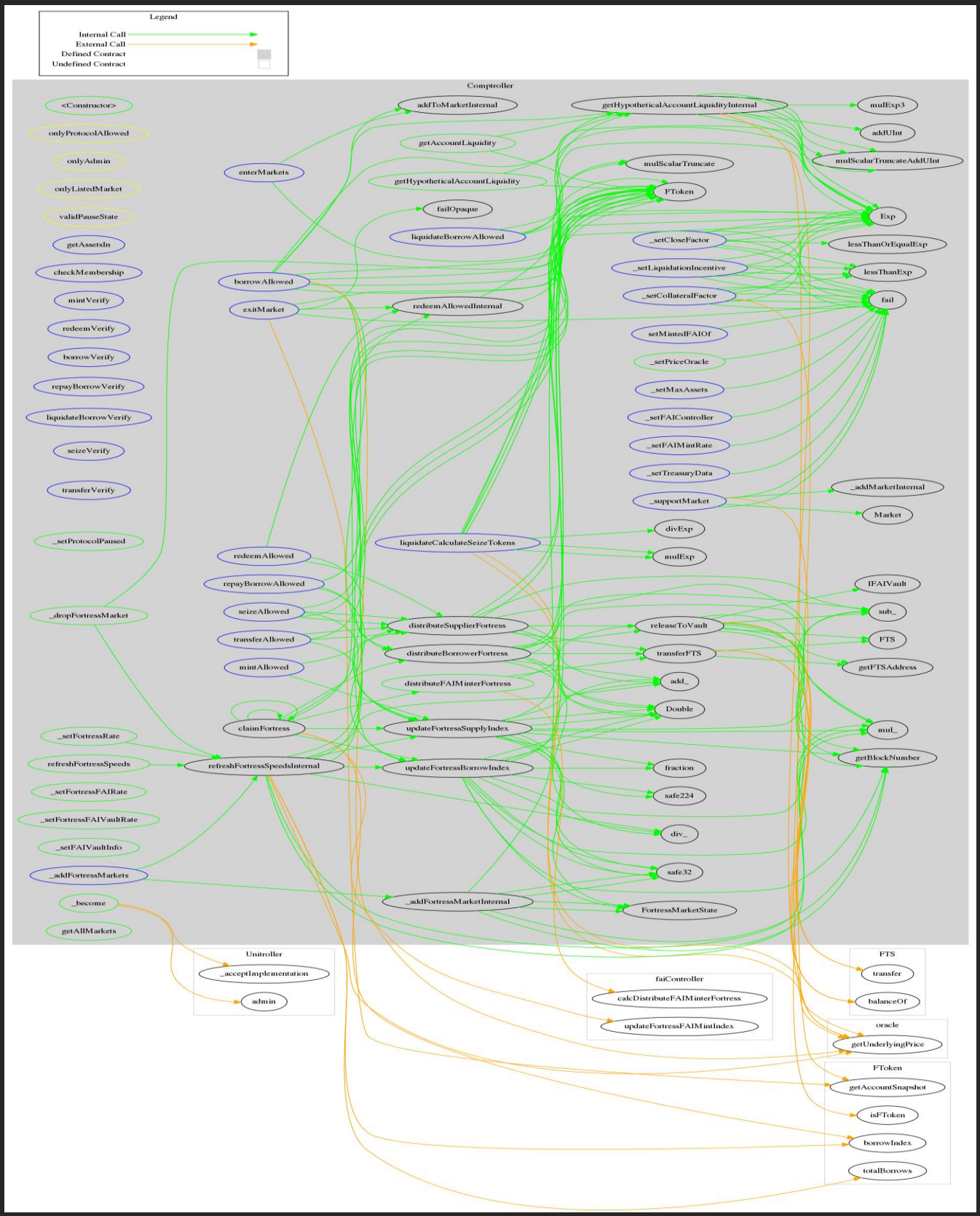
Sl.	Function	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	setDelay	write	caller was required to be contract itself	It should be an admin
3	acceptAdmin	write	Passed	No Issue
4	setPendingAdmin	write	caller was required to be contract itself	It should be an admin
5	queueTransaction	write	Passed	No Issue
6	cancelTransaction	write	Passed	No Issue
7	executeTransaction	write	Passed	No Issue
8	getBlockTimestamp	read	Passed	No Issue

GovernorAlpha.sol

Sl.	Function	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	propose	write	Passed	No Issue
3	queue	write	Passed	No Issue
4	_queueOrRevert	internal	Passed	No Issue
5	execute	write	Infinite loop possibility	Keep array length limited
6	cancel	write	Infinite loop possibility	Keep array length limited
7	getActions	read	Passed	No Issue
8	getReceipt	read	Passed	No Issue
9	state	read	Passed	No Issue
10	castVote	write	Passed	No Issue
11	castVoteBySig	write	Passed	No Issue
12	_castVote	internal	Passed	No Issue
13	__acceptAdmin	write	Passed	No Issue
14	__abdicate	write	Passed	No Issue
15	__queueSetTimelock PendingAdmin	write	Passed	No Issue
16	__executeSetTimelock PendingAdmin	write	Passed	No Issue
17	add256	read	Passed	No Issue
18	sub256	read	Passed	No Issue
19	getChainId	read	Passed	No Issue

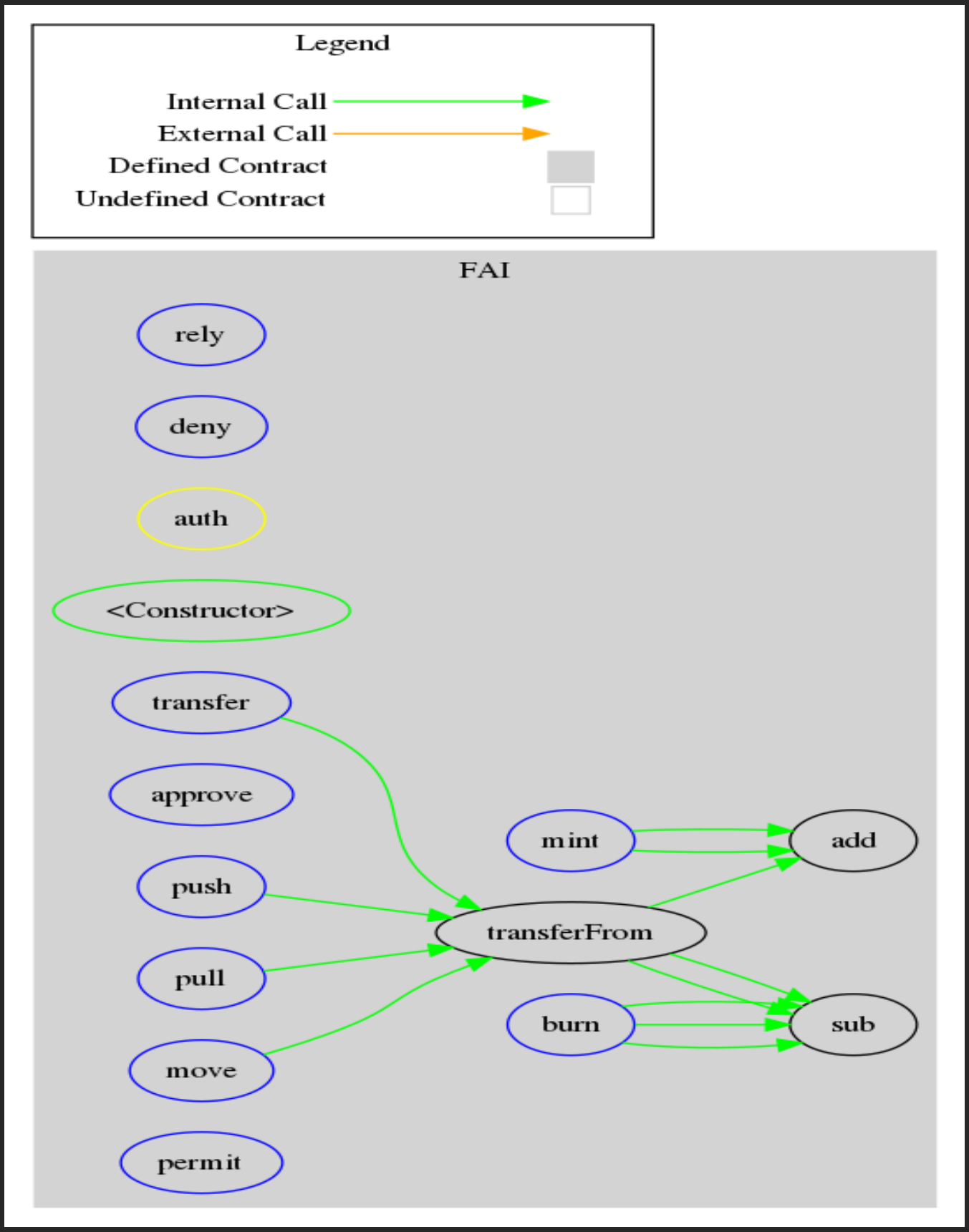
Code Flow Diagrams

Contract Comptroller



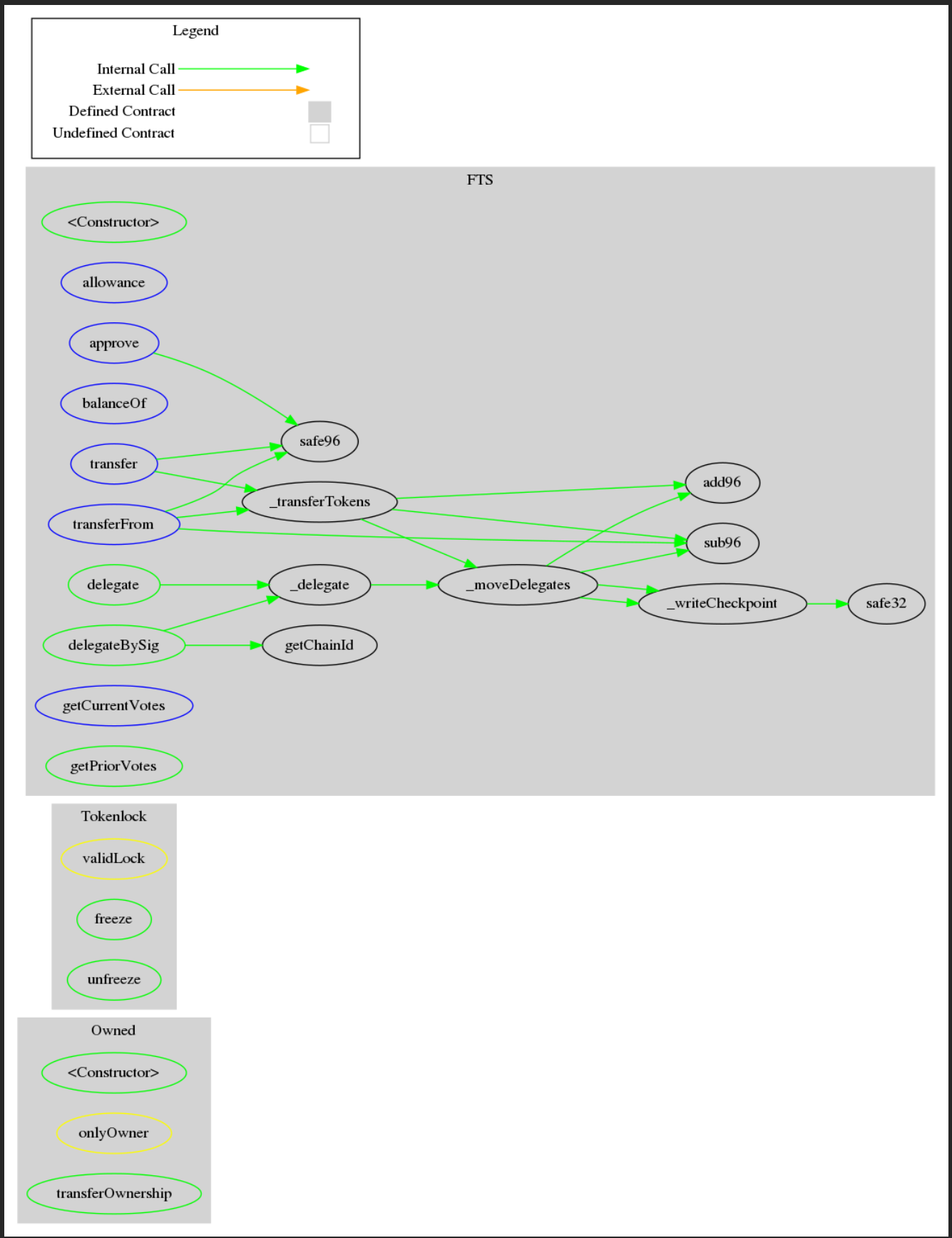
Copyright © 2021 Hash0X (audit@hash0x.com)

Contract FAI



Copyright © 2021 Hash0X (audit@hash0x.com)

Contract FTS



Slither Results Log

```
root@fedrik-ThinkPad-T410:/home/fedrik/slither# slither ../Fortress-contracts/contracts/
Compilation warnings/errors on ../Fortress-contracts/contracts/FortressPriceOracle.sol:
../Fortress-contracts/contracts/FortressPriceOracle.sol:2:1: Warning: Experimental features
are turned on. Do not use experimental features on live deployments.
pragma experimental ABIEncoderV2;
^-------------------------------^

Compilation warnings/errors on ../Fortress-contracts/contracts/GovernorAlpha.sol:
../Fortress-contracts/contracts/GovernorAlpha.sol:2:1: Warning: Experimental features are
turned on. Do not use experimental features on live deployments.
pragma experimental ABIEncoderV2;
^-------------------------------^

Compilation warnings/errors on ../Fortress-contracts/contracts/FAIVault.sol:
../Fortress-contracts/contracts/FAIVaultProxy.sol:6:1: Error: Identifier already declared.
contract FAIVaultProxy is FAIVaultAdminStorage, FAIVaultErrorReporter {
^ (Relevant source part starts here and spans across multiple lines).
../Fortress-contracts/contracts/FAIVaultProxy.sol:3:1: The previous declaration is here:
import "./FAIVaultStorage.sol";
^-----------------------------^

Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/crytic_compile/platform/solc.py", line 409, in
_run_solc
    ret = json.loads(stdout)
  File "/usr/lib/python3.6/json/__init__.py", line 354, in loads
    return _default_decoder.decode(s)
  File "/usr/lib/python3.6/json/decoder.py", line 339, in decode
    obj, end = self.raw_decode(s, idx=_w(s, 0).end())
  File "/usr/lib/python3.6/json/decoder.py", line 357, in raw_decode
    raise JSONDecodeError("Expecting value", s, err.value) from None
json.decoder.JSONDecodeError: Expecting value: line 1 column 1 (char 0)

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/slither_analyzer-0.7.1-
py3.6.egg/slither/__main__.py", line 721, in main_impl
    ) = process_all(filename, args, detector_classes, printer_classes)
  File "/usr/local/lib/python3.6/dist-packages/slither_analyzer-0.7.1-
py3.6.egg/slither/__main__.py", line 71, in process_all
    compilations = compile_all(target, **vars(args))
  File "/usr/local/lib/python3.6/dist-packages/crytic_compile/crytic_compile.py", line 1097, in
compile_all
    compilations.append(CryticCompile(filename, **kwargs))
  File "/usr/local/lib/python3.6/dist-packages/crytic_compile/crytic_compile.py", line 137, in init
```

```
    self._compile(**kwargs)
  File "/usr/local/lib/python3.6/dist-packages/crytic_compile/crytic_compile.py", line 987, in
_compile
    self._platform.compile(self, **kwargs)
  File "/usr/local/lib/python3.6/dist-packages/crytic_compile/platform/solc.py", line 108, in
compile
    targets_json = _get_targets_json(crytic_compile, self._target, **kwargs)
  File "/usr/local/lib/python3.6/dist-packages/crytic_compile/platform/solc.py", line 217, in
_get_targets_json
    force_legacy_json=force_legacy_json,
  File "/usr/local/lib/python3.6/dist-packages/crytic_compile/platform/solc.py", line 413, in
_run_solc
    raise InvalidCompilation(f"Invalid solc compilation {stderr}")
crytic_compile.platform.exceptions.InvalidCompilation: Invalid solc compilation ../Fortress-
contracts/contracts/FAIVaultProxy.sol:6:1: Error: Identifier already declared.
contract FAIVaultProxy is FAIVaultAdminStorage, FAIVaultErrorReporter {
^ (Relevant source part starts here and spans across multiple lines).
../Fortress-contracts/contracts/FAIVaultProxy.sol:3:1: The previous declaration is here:
import "./FAIVaultStorage.sol";


ERROR:root:None
ERROR:root:Error in ../Fortress-contracts/contracts/
ERROR:root:Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/crytic_compile/platform/solc.py", line 409, in
_run_solc
    ret = json.loads(stdout)
  File "/usr/lib/python3.6/json/__init__.py", line 354, in loads
    return _default_decoder.decode(s)
  File "/usr/lib/python3.6/json/decoder.py", line 339, in decode
    obj, end = self.raw_decode(s, idx=_w(s, 0).end())
  File "/usr/lib/python3.6/json/decoder.py", line 357, in raw_decode
    raise JSONDecodeError("Expecting value", s, err.value) from None
json.decoder.JSONDecodeError: Expecting value: line 1 column 1 (char 0)

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/slither_analyzer-0.7.1-
py3.6.egg/slither/__main__.py", line 721, in main_impl
    ) = process_all(filename, args, detector_classes, printer_classes)
  File "/usr/local/lib/python3.6/dist-packages/slither_analyzer-0.7.1-
py3.6.egg/slither/__main__.py", line 71, in process_all
    compilations = compile_all(target, **vars(args))
  File "/usr/local/lib/python3.6/dist-packages/crytic_compile/crytic_compile.py", line 1097, in
compile_all
    compilations.append(CryticCompile(filename, **kwargs))
  File "/usr/local/lib/python3.6/dist-packages/crytic_compile/crytic_compile.py", line 137, in init
    self._compile(**kwargs)
```

```
  File "/usr/local/lib/python3.6/dist-packages/crytic_compile/crytic_compile.py", line 987, in
_compile
    self._platform.compile(self, **kwargs)
  File "/usr/local/lib/python3.6/dist-packages/crytic_compile/platform/solc.py", line 108, in
compile
    targets_json = _get_targets_json(crytic_compile, self._target, **kwargs)
  File "/usr/local/lib/python3.6/dist-packages/crytic_compile/platform/solc.py", line 217, in
_get_targets_json
    force_legacy_json=force_legacy_json,
  File "/usr/local/lib/python3.6/dist-packages/crytic_compile/platform/solc.py", line 413, in
_run_solc
    raise InvalidCompilation(f"Invalid solc compilation {stderr}")
crytic_compile.platform.exceptions.InvalidCompilation: Invalid solc compilation ../Fortress-
contracts/contracts/FAIVaultProxy.sol:6:1: Error: Identifier already declared.
contract FAIVaultProxy is FAIVaultAdminStorage, FAIVaultErrorReporter {
^ (Relevant source part starts here and spans across multiple lines).
../Fortress-contracts/contracts/FAIVaultProxy.sol:3:1: The previous declaration is here:
import "./FAIVaultStorage.sol";
```