FRACTAL

FRACTAL

SMART CONTRACT AUDIT

zokyo

July 28th 2022 | v. 1.0

Security Audit Score

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

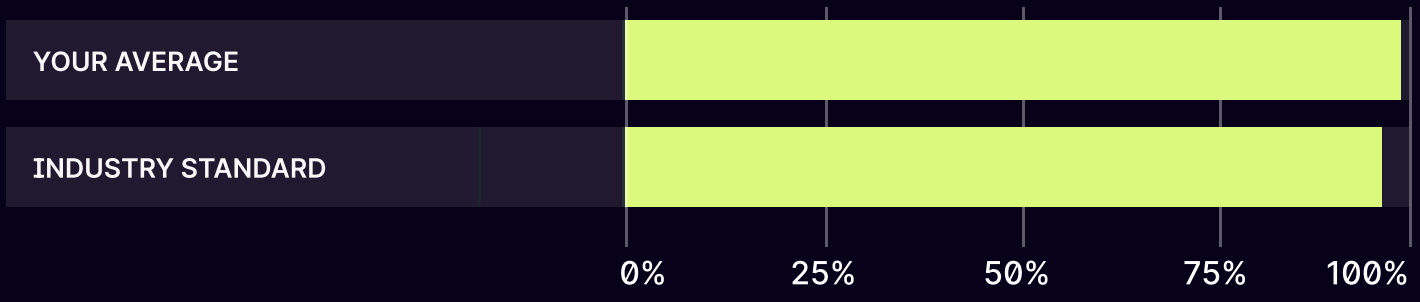
TECHNICAL SUMMARY

This document outlines the overall security of the Fractal smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the Fractal smart contract codebase for quality, security, and correctness.

Contract Status

Testable Code



The testable code is 96%, which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Zokyo recommend that the Fractal team put in place a bug bounty program to encourage further and active analysis of the smart contract.

Table of Contents

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the Fractal repositoritories.

Repository: https://github.com/decent-dao/fractal-contracts

Initial commit: e06072c4b2ccd571939e56f78accb96b3832e56f

Last audited commit: d9832d3076fdd7f907ef19444e150bb2eb30d327

https://github.com/decent-dao/fractal-metafactory

Initial commit: 7ab0016d8c20fa1da2bfcc82086c5579abc8e2e0

Last audited commit: 705317f48bb91bcbe2d9686b025e36b2546fe57a

https://github.com/decent-dao/fractal-module-treasury

Initial commit: a810c36ed87c66c65660379dd277fbb3a36467ad

Last audited commit: fde4ac32ebcb1bf3d51c26e17073338e59aebe59

https://github.com/decent-dao/fractal-module-governor

Initial commit: 14502c14c985a03e68514c0cf3c39f0a242348fe

Last audited commit: 989bbc2c995d88e4381a47911a13b972a0e457d9

https://github.com/decent-dao/fractal-votes-token

Initial commit: 84472e5267bec5195d14660bc75ed91e49fe2a16

Last audited commit: 7f98c9c9fb917cd73652b9d2a6116ad011d79cc8

Within the scope of this audit Zokyo auditors have reviewed the following contract(s):

- DAO.sol
- DAOAccessControl.sol
- DAOFactory.sol
- ModuleBase.sol
- ModuleFactoryBase.sol
- TreasuryModule.sol
- TreasuryModuleFactory.sol
- MetaFactory.sol
- Timelock.sol
- TokenFactory.sol
- VotesToken.sol
- VotesTokenWithSupply.sol
- GovernorFactory.sol
- GovernorModule.sol
- GovernorTimelock.sol

Throughout the review process, care was taken to ensure that the contract:

- Implements and adheres to existing standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of resources, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of Fractal smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

01	Due diligence in assessing the overall code quality of the codebase.
02	Cross-comparison with other, similar smart contracts by industry leaders.
03	Testing contract logic against common and uncommon attack vectors.
04	Thorough, manual review of the codebase, line-by-line.

Executive Summary

The Fractal protocol is a set of basic contracts combined into one ecosystem. There is a DAO contract in which the basic settings for managing roles are initialized and it is sequentially available to all other necessary contracts in the protocol. There are three separate modules - the storage module (TreasuryModule), which allows you to receive / store funds of ETH, ERC20, ERC721 tokens and also send them to specified addresses. There is a voting module (GovernorModule) which, in cooperation with delay modules (Timelock, GovermorTimelock), implements voting for users who own VotesToken (third module).

Each module, both the DAO itself and globally for the entire protocol, has separate factory contracts that allow one transaction to create and initialize all the necessary contracts for the module.

The protocol does not have complex business logic, however, it requires special attention during deployment, since many elements depend on the correctness of the specified parameters, such as calldata for deployment. This poses an increased risk for human factor vulnerability.

During the review of the contract, we found:

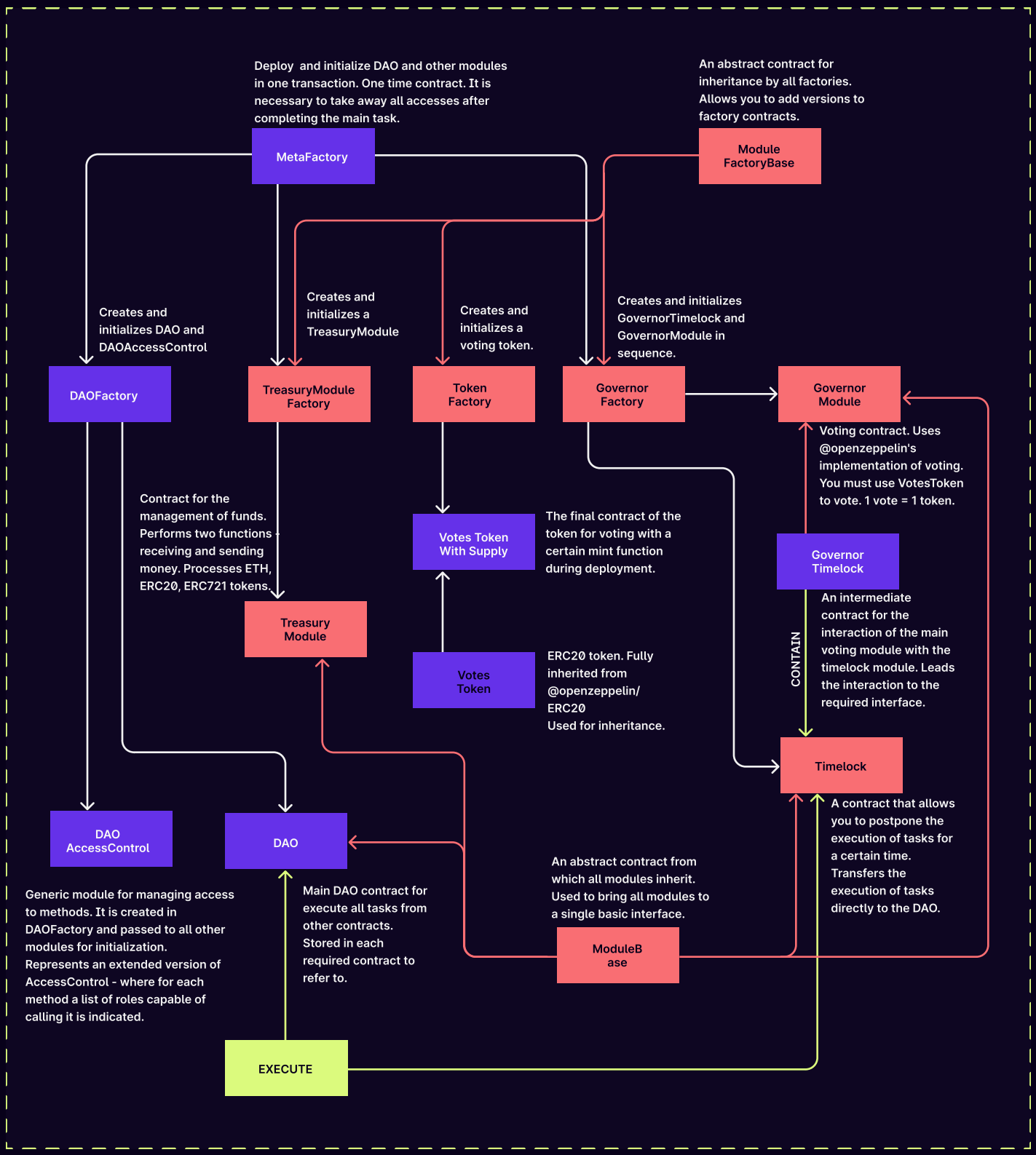
1) errors that would lead to incorrect operation of the protocol

2) non-working elements of logic

3) places recommended for correction for better gas optimization.

Our team has found all of the above vulnerabilities and described possible solutions. From the point of view of security, no violations were found, most of the errors found were minor, related to typos and were more of an informational nature. The verification was carried out by manually going through and fully analyzing all the contracts of the project together with automated tools.

PROTOCOL OVERVIEW



STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Issues tagged “Verified” contain unclear or suspicious functionality that either needs explanation from the Customer’s side or it is an issue that the Customer disregards as an issue. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

HIGH | RESOLVED

Duplicate of increment.
TreasuryModule function withdrawERC20Tokens

```
100:    for (uint256 index = 0; index < tokenAddressesLength; index ++) {
105:      unchecked {
106:        index ++;
107:      }
```

In one cycle, the index is incremented twice.

Recommendation:
Remove increment from for head part.

HIGH | VERIFIED

Non-minteble token.
VotesToken.sol , VotesTokenWithSupply.sol
This token does not have external mint and burn functions. That is, when creating it, the number of tokens will not change.
Even if you use VotesTokenWithSupply - in addition to the initial amount, over time you cannot influence the number of tokens in any way.

Recommendation:
Consider extending the functionality of mint and burn. Perhaps at least give a similar opportunity to the role of an owener, or something like that.

From client:
This isn't a bug, but more of a design decision. Fractal team has decided for governance tokens to have a fixed supply that cannot be changed, and to keep this code as is.

MEDIUM | RESOLVED

Invalid auth require

Timelock.sol

```
46:     function updateDelay(uint256 newDelay) external virtual authorized {
47:         require(
48:             msg.sender == address(this),
49:             "TimelockController: caller must be timelock"
50:         );
51:         emit MinDelayChange(minDelay, newDelay);
52:         minDelay = newDelay;
53:     }
```

This method cannot be executed. Because the method has a check that it can only be called by the contract itself. No calls to this method were found in the rest of the code. Also, in the entire protocol, it was not found possible to execute this method on behalf of this contract.

Recommendation:

Remove the check, a basic authorization modifier should suffice. Or reconsider the logic in this place.

LOW | RESOLVED

Duplicate.

File: DAOAccessControl.sol function initialize

```
45:             targets.length != functionDescs.length ||
46:             targets.length != functionDescs.length
```

Same check.

Recommendation:

Remove the duplicate code, or replace the variable in the second check (if it was planned so)

From client:

This is a bug that will be fixed

Unused variable.

VotesTokenWithSupply.sol function constructor

```
24:         uint256 tokenSum;
25:         for (uint256 i = 0; i < hodlers.length; i++) {
26:             _mint(hodlers[i], allocations[i]);
27:             tokenSum += allocations[i];
28:         }
29:     }
```

You are processing tokenSum variable which is then not used anywhere

Recommendation:

Remove tokenSum variable.

From client:

This is a bug that will be fixed

Not required array
TokenFactory.sol , TreasuryModuleFactory.sol, function create
In these functions, when calling Create2.deploy, the result of the call is put into an array with one element. This is not necessary, and from a gas optimization point of view, it is slightly cheaper to accept and return a simple address type.

Recommendation:
Replace address[1] with address.

From client:
The GovernorFactory (and potentially other future modules) create two contracts, and return two addresses, which is why the array is needed

https://github.com/decent-dao/fractal-module-governor/blob/main/contracts/Governor/GovernorFactory.sol#L31-L32

Since all factories should use this same create function interface, Fractal team has decided to keep this code as is

Never used.
ModuleFactoryBase.__initFactoryBase()
This function is never used.

Recommendation:
Remove the function.

From client:
This function is actually used by both TreasuryModuleFactory.sol and the GovernorFactory.sol, and will likely be used by future module factories.

https://github.com/decent-dao/fractal-module-treasury/blob/main/contracts/TreasuryModuleFactory.sol#L22

https://github.com/decent-dao/fractal-module-governor/blob/main/contracts/Governor/GovernorFactory.sol#L17

Fractal team has decided to keep this code as is

No checking balance

File: Timelock.sol

```
250:    function _beforeCall(bytes32 id, bytes32 predecessor) private view {
251:        require(
252:            isOperationReady(id),
253:            "TimelockController: operation is not ready"
254:        );
255:        require(
256:            predecessor == bytes32(0) || isOperationDone(predecessor),
257:            "TimelockController: missing dependency"
258:        );
259:    }
```

Formally - there are not correct checks. Since it first checks "whether the task is ready", and then whether it is "completed". We understand that there will be no error in this case, since this module will be used through GovTimelock. But for independent use - it needs to be fixed.

Recommendation:

Add a negation to the isOperationDone check

From client:

After reviewing this, Fractal team believes there is no logical issue here. Note that “isOperationReady” is being passed “id”, while “isOperationDone” is being passed “predecessor”.

	DAO.sol	DAOAccessControl.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	DAOFactory.sol	ModuleBase.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	ModuleFactoryBase.sol	TokenFactory.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	VotesToken.sol	VotesTokenWithSupply.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	MetaFactory.sol	TreasuryModule.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	TreasuryModuleFactory.sol	GovernorFactory.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

	GovernorModule.sol	GovernorTimelock.sol
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

Timelock.sol	
Re-entrancy	Pass
Access Management Hierarchy	Pass
Arithmetic Over/Under Flows	Pass
Unexpected Ether	Pass
Delegatecall	Pass
Default Public Visibility	Pass
Hidden Malicious Code	Pass
Entropy Illusion (Lack of Randomness)	Pass
External Contract Referencing	Pass
Short Address/ Parameter Attack	Pass
Unchecked CALL Return Values	Pass
Race Conditions / Front Running	Pass
General Denial Of Service (DOS)	Pass
Uninitialized Storage Pointers	Pass
Floating Points and Precision	Pass
Tx.Origin Authentication	Pass
Signatures Replay	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Secured team

As part of our work assisting Fractal in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Truffle testing framework.

Tests were based on the functionality of the code, as well as a review of the Fractal contract requirements for details about issuance amounts and how the system handles these.

FRACTAL_META_FACTORY

MetaFactory

- ✓ Should revert create dao if unequal arrays lengths module factories bytes (422ms)
- ✓ Should revert create dao if unequal arrays lengths (701ms)
- ✓ Should sets up the correct Governor module action authorization (44ms)
- ✓ Should sets up the correct timelock action authorization (41ms)
- ✓ Should allocated correct token amounts
- ✓ Should supports creating, voting on, and executing a proposal (409ms)

6 passing

FRACTAL_CORE

DAO Access Control Contract Audit

Initilize Access Control

- ✓ should revert initialize if Unequal Array Lengths (44ms)
- ✓ Should setup Executor Role (38ms)

Roles

- ✓ Should reverted daoGrantRoles if unequal arrays length
- ✓ Admin of a role can grant new members with daoGrantRoles (123ms)
- ✓ should revert daoGrantRolesAndAdmins if unequl arrays length roles
- ✓ should revert daoGrantRolesAndAdmins if unequl arrays length members
- ✓ Admin of a role can grant new members (254ms)
- ✓ Non-Admin of a role can not grant new members (177ms)
- ✓ Admin of a role can revoke members (510ms)
- ✓ Non-Admin of a role can not revoke members (192ms)
- ✓ A role member can renounce their role (422ms)
- ✓ A role member cannot renounce another user's role (156ms)
- ✓ Should batch create Roles (292ms)
- ✓ Should override/update Role Admins (254ms)
- ✓ Should revert UnAuthorized (batch create)

Other Tests

```
    ✓ should return true if role authorized (144ms)
    ✓ should return false if role non-authorized
    ✓ Should revert Non-Authorized User (Add)
    ✓ Should revert if non equal arrays length targets
    ✓ Should revert if non equal arrays length targets
    ✓ Should dont Remove Actions if user dont have a role (194ms)
    ✓ Should revert remove actions if UnequalArrayLengths
    ✓ Should revert remove actions if UnequalArrayLengths targets and roles
    ✓ Should remove unauthorized user (106ms)
  DAO Audit
  execute dao
    ✓ Should revert execute if unequal array lengths
  25 passing
FRACTAL_GOVER_MODULE
  Gov Module Audit
  Quourum
    ✓ Should revert if porposal not successeful (305ms)
    ✓ Should revert if the delay is not successful (1084ms)
  Execute proposal
    ✓ Should execute a passing proposal (4088ms)
  Execute timelock methods
    ✓ Should execute update delay (1416ms)
    ✓ Should return eta for executing propose
    ✓ Should revert update delay if caller, not a timelock
    ✓ Should revert scheduleBatch if values array length, not equal targets lengths (238ms)
    ✓ Should revert scheduleBatch if datas array length, not equal targets lengths (215ms)
    ✓ Should revert executeBatch if values array length, not equal targets lengths (200ms)
    ✓ Should revert executeBatch if datas array length not equal targets lengths (218ms)
    ✓ Should cancel proposal (691ms)
  10 passing, 1 failed
```

FILE	% STMTS	% BRANCH	% FUNCS
DAO.sol	100	100	100
DAOAccessControl.sol	100	92.63	95.83
DAOFactory.sol	100	100	100
ModuleBase.sol	100	100	100
ModuleFactoryBase.sol	100	100	100
TokenFactory.sol	100	100	100
VotesToken.sol	75	100	75
VotesTokenWithSupply.sol	100	100	100
MetaFactory.sol	100	100	100
TreasuryModule.sol	100	100	100
TreasuryModuleFactory.sol	100	100	100
GovernorFactory.sol	100	100	100
GovernorModule.sol	91.67	100	93.67
GovernorTimelock.sol	80.75	75	77.34
Timelock.sol	93	75	92.68
All files	**96.028**	**96.17**	**95.63**

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Fractal team

As part of our work assisting Fractal team in verifying the correctness of their contract code, our team has checked the complete set of unit tests prepared by the Fractal team.

It needs to be mentioned, that the original code has a significant original coverage with testing scenarios provided by the Fractal team. All of them were also carefully checked by the auditors' team.

FRACTAL_CORE_CONTRACTS

DAO Access Control Contract

Initilize Access Control

- ✓ Supports the expected ERC165 interface
- ✓ Should setup Executor Role
- ✓ Should setup Default Admin Role (48ms)
- ✓ Should setup Roles (53ms)
- ✓ Supports the Open Role (53ms)

Roles

- ✓ Admin of a role can grant new members (543ms)
- ✓ Non-Admin of a role can not grant new members (367ms)
- ✓ Admin of a role can revoke members (1349ms)
- ✓ Non-Admin of a role can not revoke members (381ms)
- ✓ A role member can renounce their role (978ms)
- ✓ A role member cannot renounce another user's role (331ms)
- ✓ Should batch create Roles (410ms)
- ✓ Should override/update Role Admins (681ms)
- ✓ Should revert UnAuthorized (batch create) (40ms)

Action Roles

- ✓ Should setup Actions (400ms)
- ✓ Should revert Non-Authorized User (Add)
- ✓ Should Remove Actions (1195ms)
- ✓ Unauthorized user (Remove) (221ms)

DAO

an empty DAO

- ✓ Supports the expected ERC165 interface
- ✓ has the DAO_ROLE
- ✓ doesn't allow anyone to grant the EXECUTE role
- ✓ doesn't allow anyone to revoke existing roles

```
    ✓ doesn't allow anyone to call `execute` (50ms)
  a dao with `execute` permissions
    ✓ Init Access Control
    ✓ Init Dao Name
    ✓ executor EOA should be able to call `execute` (218ms)
    ✓ UnAuthUser should NOT be able to call `execute`
    ✓ UnAuthDAO should NOT be able to call `execute` (576ms)
  DAOFactory
    ✓ emits an event with the new DAO's address
    ✓ sets up moduleBase
    ✓ Can predict DAO and Access Control
    ✓ Base Init for DAO
    ✓ Base Init for Access Control (44ms)
    ✓ Executor Role is set (48ms)
    ✓ Upgrade Role is set (43ms)
    ✓ Should setup Actions
    ✓ Revert Initilize
    ✓ executor EOA should be able to call `execute` (826ms)
    ✓ Non executor EOA should revert w/ Execute (41ms)
    ✓ upgrader EOA should be able to upgrade (809ms)
    ✓ Non-upgrader should revert w/ UpgradeTo (82ms)
    ✓ Supports the expected ERC165 interface
FRACTAL_VOTES_TOKEN
  Token Factory
  Token / Factory
    ✓ Token/Factory Deployed
    ✓ Can predict Token Address
    ✓ Init is correct
    ✓ Balances are correct
    ✓ Token Factory does not deploy with incorrect data
    ✓ Supports the expected ERC165 interface
  6 passing (17s)
FRACTAL_META_FACTORY
  MetaFactory
    ✓ Emitted events with expected deployed contract addresses
    ✓ Setup the correct roles (136ms)
    ✓ Sets up the correct DAO role authorization (123ms)
    ✓ Sets up the correct Treasury role authorization (238ms)
    ✓ Sets up the correct Governor module role authorization (39ms)
    ✓ Sets up the correct timelock role authorization (330ms)
    ✓ Sets up the correct DAO action authorization (70ms)
    ✓ Sets up the correct Treasury action authorization (203ms)
```

- ✓ Sets up the correct Governor module action authorization
- ✓ Sets up the correct timelock action authorization
- ✓ Supports the expected ERC165 interface
- ✓ Allocated correct token amounts
- ✓ Supports creating, voting on, and executing a proposal (403ms)

13 passing (25s)

FRACTAL_MODULE_TREASURY

Treasury

Treasury supports Ether

- ✓ Returns the module name
- ✓ Supports the expected ERC165 interfaces
- ✓ Receives Ether
- ✓ Emits an event when ETH is withdrawn (75ms)
- ✓ Sends Eth using the withdraw function (74ms)
- ✓ Sends ETH to multiple addresses using the withdraw function (68ms)
- ✓ Reverts when a non-owner attempts to withdraw ETH (53ms)
- ✓ Reverts when the withdraw function is called with inequal array lengths (38ms)

Treasury supports ERC-20 tokens

- ✓ Receives ERC-20 tokens (68ms)
- ✓ Emits event when ERC-20 tokens are deposited (100ms)
- ✓ Receives ERC-20 tokens using the deposit function (99ms)
- ✓ Receives multiple ERC-20 tokens from multiple addresses using the deposit function (350ms)
- ✓ Emits event when ERC-20 tokens are withdrawn (76ms)
- ✓ Sends ERC-20 tokens using the withdraw function (95ms)
- ✓ Sends multiple ERC-20 tokens to multiple addresses using the withdraw function (337ms)
- ✓ Reverts when a non authorized user attempts to withdraw ERC-20 tokens
- ✓ Reverts when the deposit function is called with inequal array lengths (59ms)
- ✓ Reverts when the withdraw function is called with inequal array lengths (61ms)

Treasury supports ERC-721 tokens

- ✓ Receives ERC-721 tokens (62ms)
- ✓ Emits an event when ERC-721 tokens are deposited (343ms)
- ✓ Receives ERC-721 tokens using the deposit function (323ms)
- ✓ Receives multiple ERC-721 tokens from multiple addresses using the deposit function (1276ms)
- ✓ Emits an event when ERC-721 tokens are withdrawn (82ms)
- ✓ Sends ERC-721 tokens using the withdraw function (80ms)
- ✓ Sends multiple ERC-721 tokens to multiple addresses using the withdraw function (773ms)
- ✓ Reverts when a non-owner attempts to withdraw ERC-721 tokens (41ms)
- ✓ Reverts when the deposit function is called with inequal array lengths (56ms)
- ✓ Reverts when the withdraw function is called with inequal array lengths (59ms)

Treasury Factory

ModuleFactoryBase

- ✓ New version can be added to the version Control

- ✓ New version cannot be added by an unauthorized user
- ✓ Returns current version
- ✓ treasury returns correct factory
- ✓ Can predict DAO and Access Control

Supports authorized upgradeability
- ✓ Can be upgraded by an authorized user (184ms)
- ✓ Cannot be upgraded by an unauthorized user (83ms)

Treasury supports Ether
- ✓ Supports the expected ERC165 interfaces
- ✓ Receives Ether
- ✓ Emits an event when ETH is withdrawn (182ms)
- ✓ Sends Eth using the withdraw function (183ms)
- ✓ Sends ETH to multiple addresses using the withdraw function (202ms)
- ✓ Reverts when a non-owner attempts to withdraw ETH
- ✓ Reverts when the withdraw function is called with inequal array lengths (44ms)

Treasury supports ERC-20 tokens
- ✓ Receives ERC-20 tokens (47ms)
- ✓ Emits event when ERC-20 tokens are deposited (312ms)
- ✓ Receives ERC-20 tokens using the deposit function (312ms)
- ✓ Receives multiple ERC-20 tokens from multiple addresses using the deposit function (1241ms)
- ✓ Emits event when ERC-20 tokens are withdrawn (275ms)
- ✓ Sends ERC-20 tokens using the withdraw function (288ms)
- ✓ Sends multiple ERC-20 tokens to multiple addresses using the withdraw function (1126ms)
- ✓ Reverts when a non authorized user attempts to withdraw ERC-20 tokens
- ✓ Reverts when the deposit function is called with inequal array lengths (60ms)
- ✓ Reverts when the withdraw function is called with inequal array lengths (58ms)

Treasury supports ERC-721 tokens
- ✓ Receives ERC-721 tokens (62ms)
- ✓ Emits an event when ERC-721 tokens are deposited (333ms)
- ✓ Receives ERC-721 tokens using the deposit function (346ms)
- ✓ Receives multiple ERC-721 tokens from multiple addresses using the deposit function (1393ms)
- ✓ Emits an event when ERC-721 tokens are withdrawn (289ms)
- ✓ Sends ERC-721 tokens using the withdraw function (298ms)
- ✓ Sends multiple ERC-721 tokens to multiple addresses using the withdraw function (1220ms)
- ✓ Reverts when a non-owner attempts to withdraw ERC-721 tokens
- ✓ Reverts when the deposit function is called with inequal array lengths (55ms)
- ✓ Reverts when the withdraw function is called with inequal array lengths (57ms)

62 passing (2m)

FRACTAL_GOVER_MODULE

Gov Module

Init DAO
- ✓ Contracts are deployed

- ✓ Initiate DAO
- ✓ Initiate Timelock Controller
- ✓ Gov Module (87ms)
- ✓ Gov Token (99ms)
- ✓ Supports the expected ERC165 interface (61ms)

Proposals

- ✓ Creates a DAO proposal (109ms)
- ✓ Reverts w/ out delegated vote weight (59ms)
- ✓ Reverts w/ duplicate proposal (128ms)
- ✓ Creates two DAO proposals (238ms)

Votes

- ✓ Allows voting on two DAO proposals (645ms)
- ✓ Revert voting before votes start (121ms)
- ✓ Revert proposal does not exist (40ms)
- ✓ Revert duplicate votes (526ms)
- ✓ Users without vote power do not update status (306ms)
- ✓ Users without delegate votes cannot delegate votes after voting (532ms)
- ✓ Users can delegate votes (456ms)
- ✓ Revert votes after voting period (171ms)

Queue

- ✓ Queues a passed proposal (861ms)
- ✓ Does not allow a proposal without quorum to get queued (329ms)
- ✓ Does not allow a proposal without votes to get queued (136ms)

PreventLateQuorum

- ✓ Queues proposal w/ quorum delay (465ms)
- ✓ Reverts if Quorum delay is not respected (662ms)

Execution

- ✓ Should execute a passing proposal (2451ms)
- ✓ Revert if execution is too early (732ms)
- ✓ Does not allow a non proposalid to be executed
- ✓ Does not allow a proposal to be executed before it is queued (603ms)

Gov Module Factory

ModuleFactoryBase

- ✓ sets up moduleBase
- ✓ New version can be added to the version Control (68ms)
- ✓ New version cannot be added by an unauthorized user
- ✓ Returns current version (81ms)
- ✓ Gov returns correct factory

Init Gov + timelock

- ✓ Can predict Timelock and Governor
- ✓ emits an event with the new Gov's address
- ✓ Contracts are deployed

```
    ✓ Initiate Timelock Controller (43ms)
    ✓ Gov Module (137ms)
    ✓ Supports the expected ERC165 interface
  Execution
    ✓ Should setup Roles
    ✓ Should setup Actions (94ms)
    ✓ Should execute a passing proposal (4429ms)
    ✓ Should revert if non Gov tries to execute (4374ms)
    ✓ Revert if execution is too early (2852ms)
    ✓ Revert non Gov schedules transaction (1757ms)
    ✓ Does not allow a non proposalid to be executed
    ✓ Does not allow a proposal to be executed before it is queued (1870ms)
    ✓ Should upgrade the timelock contract (4186ms)
    ✓ Should upgrade the timelock contract (4319ms)
  48 passing (4m)
```

FILE	% STMTS	% BRANCH	% FUNCS
DAO.sol	100	58.33	100
DAOAccessControl.sol	88.61	50	81.65
DAOFactory.sol	100	56.25	100
ModuleBase.sol	100	100	100
ModuleFactoryBase.sol	0	100	0
TokenFactory.sol	100	100	100
VotesToken.sol	75	100	75
VotesTokenWithSupply.sol	100	100	100
MetaFactory.sol	100	50	100
TreasuryModule.sol	100	100	100
TreasuryModuleFactory.sol	100	100	100
GovernorFactory.sol	100	100	100
GovernorModule.sol	91.67	100	86.67
GovernorTimelock.sol	68.75	66.67	63.64
Timelock.sol	80	40	81.22
All files	**86,93**	**81.41**	**83.21**

We are grateful to have been given the opportunity to work with the Fractal team.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

Zokyo's Security Team recommends that the Fractal team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo