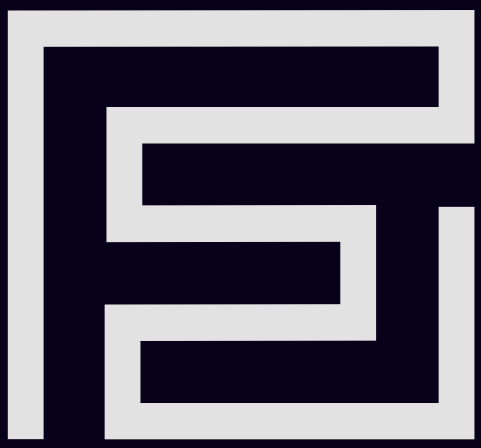


FNDZ

SMART CONTRACT AUDIT

zokyo

June, 15th, 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.

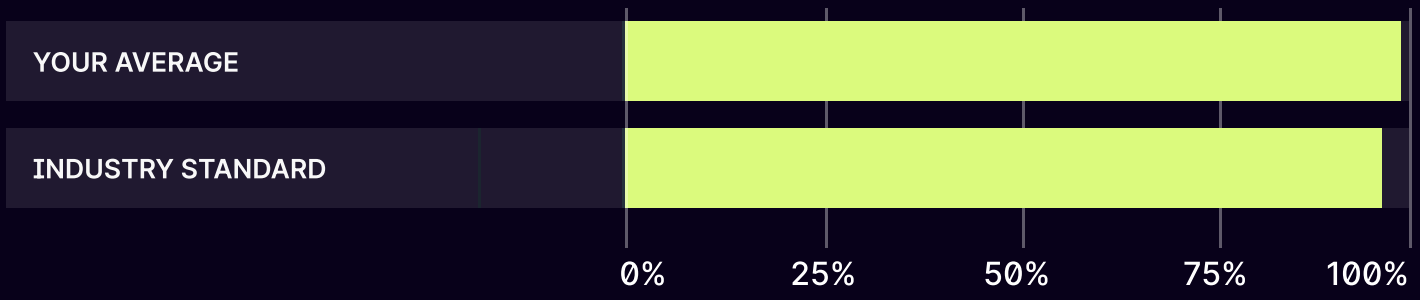
TECHNICAL SUMMARY

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

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

Contract Status

Testable Code



The testable code is 98%, 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 FNDZ 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 FNDZ repository:
https://github.com/curvegrid/fndz-core

Initial commit: 9a249cbfb84fd2dbde41e50489c3541002d6b51b

Last reviewed commit: d0b073e3165622068041e3081cc8160332c7fa58

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

- ChainlinkPriceAggregator.sol
- FNDZController.sol
- FNDZInvestmentRegistry.sol
- FNDZStaking.sol
- PriceAggregatorProxy.sol
- ReferralRegistry.sol
- Dispatcher.sol
- VaultLibBase1.sol
- VaultLibBaseCore.sol
- VaultProxy.sol
- FundDeployer.sol
- ComptrollerLib.sol
- ComptrollerProxy.sol
- VaultLib.sol
- FeeManager.sol
- EntranceReferralFee.sol
- FNDZInvestmentFee.sol
- ManagementFee.sol
- InlineSwapMixin.sol
- StakeOrderLinking.sol
- PerformanceFee.sol
- IntegrationManager.sol
- ParaSwapV5Adapter.sol
- ParaSwapV5ActionsMixin.sol
- PolicyManager.sol
- InvestorWhitelist.sol
- MinMaxInvestment.sol
- PreBuySharesValidatePolicyBase.sol
- AddressListPolicyMixin.sol
- PolicyBase.sol
- ExtensionBase.sol
- FundDeployerOwnerMixin.sol
- PermissionedVaultActionMixin.sol
- AggregatedDerivativePriceFeed.sol
- SynthetixPriceFeed.sol
- ValueInterpreter.sol
- FundActionsWrapper.sol

Throughout the review process, Zokyo Security ensures 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 FNDZ 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. 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 auditor's team has verified the whole pack of contracts. The protocol is designed for trading on AMM's and staking protocol tokens with a custom rewards system. Overall, contracts are well-structured and organized. FNDZ team has prepared a solid test suite, which is sufficient for contracts' security. It was verified that contracts correspond to the business logic of the protocol and that all the funds are safely stored. It was verified that none except users have direct access to their funds. Auditors have performed several audit rounds in order to verify the correctness of the contracts.

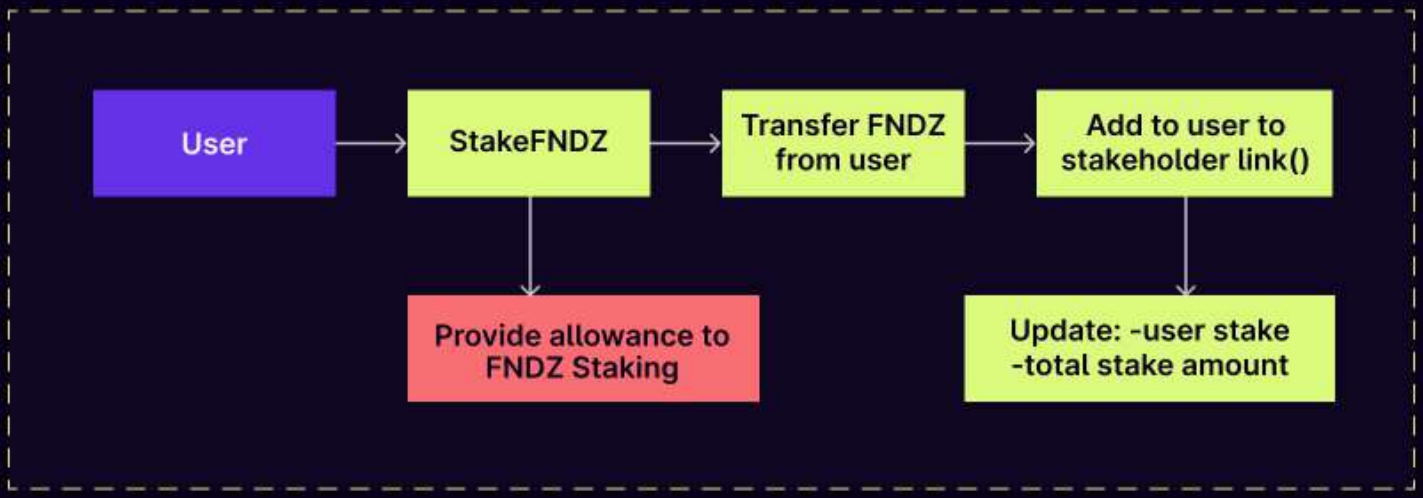
Auditors have pointed to the potential issue with the extra permission for the specific role within the contracts set. Though, after the discussion with the FNDZ team the feature was verified to be implemented according to the design. Also, FNDZ team has verified the existence of specific procedures which guarantee the safeness of the funds.

Other issues were connected to the unfinished functionality, missing checks and missed functional cases. All discovered issues are minor and don't impact the protocol's ability to operate. Nevertheless, all issues were resolved by the FNDZ team.

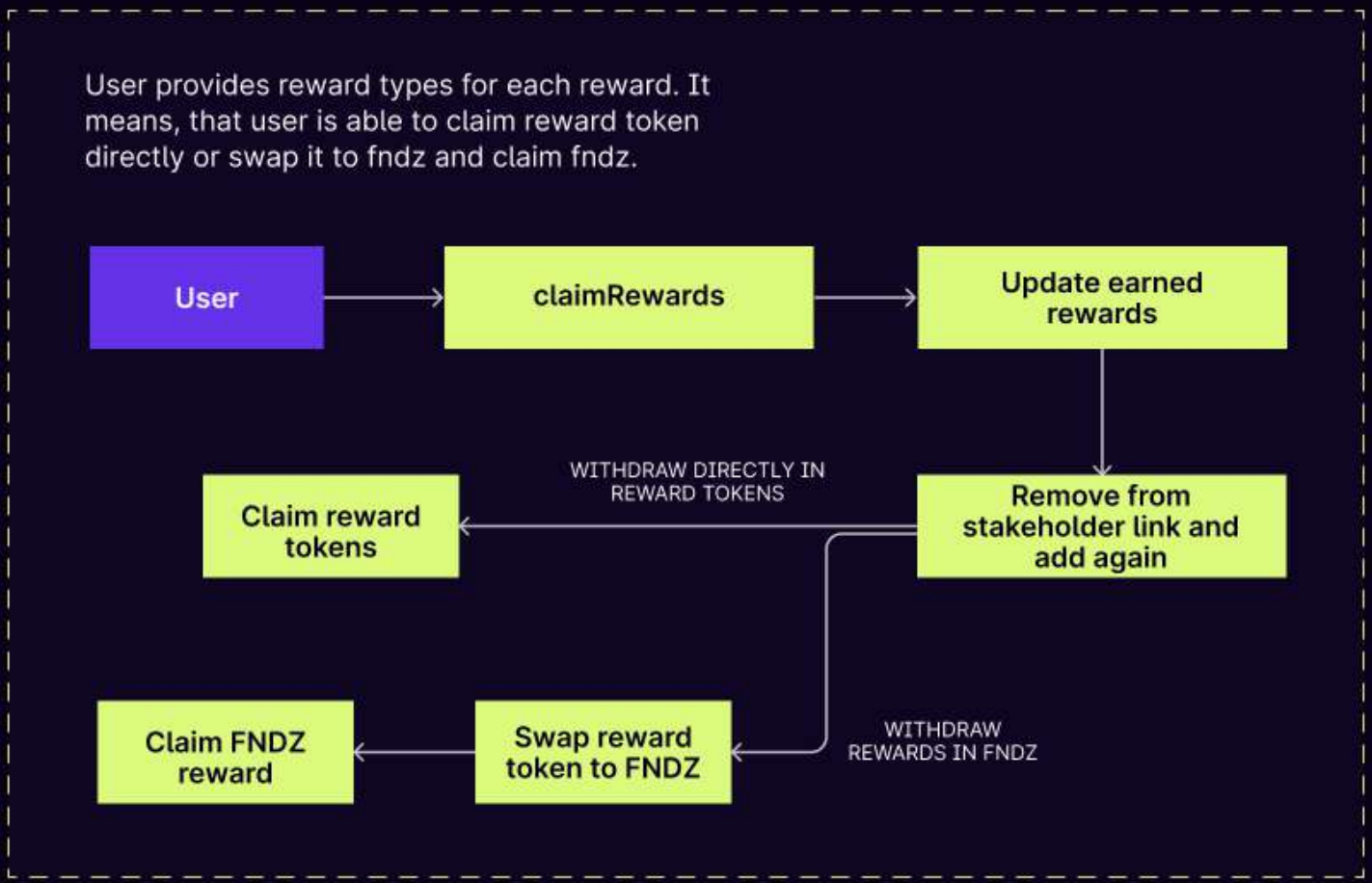
Auditors have verified native test-coverage in order to check its correctness. The team's tests are well-organized and verify all the functions of smart contracts and protocol work. Also, the auditor's team has provided its own tests and testing scenarios to confirm that contracts correspond to the security standard.

PROTOCOL OVERVIEW

1. STAKE

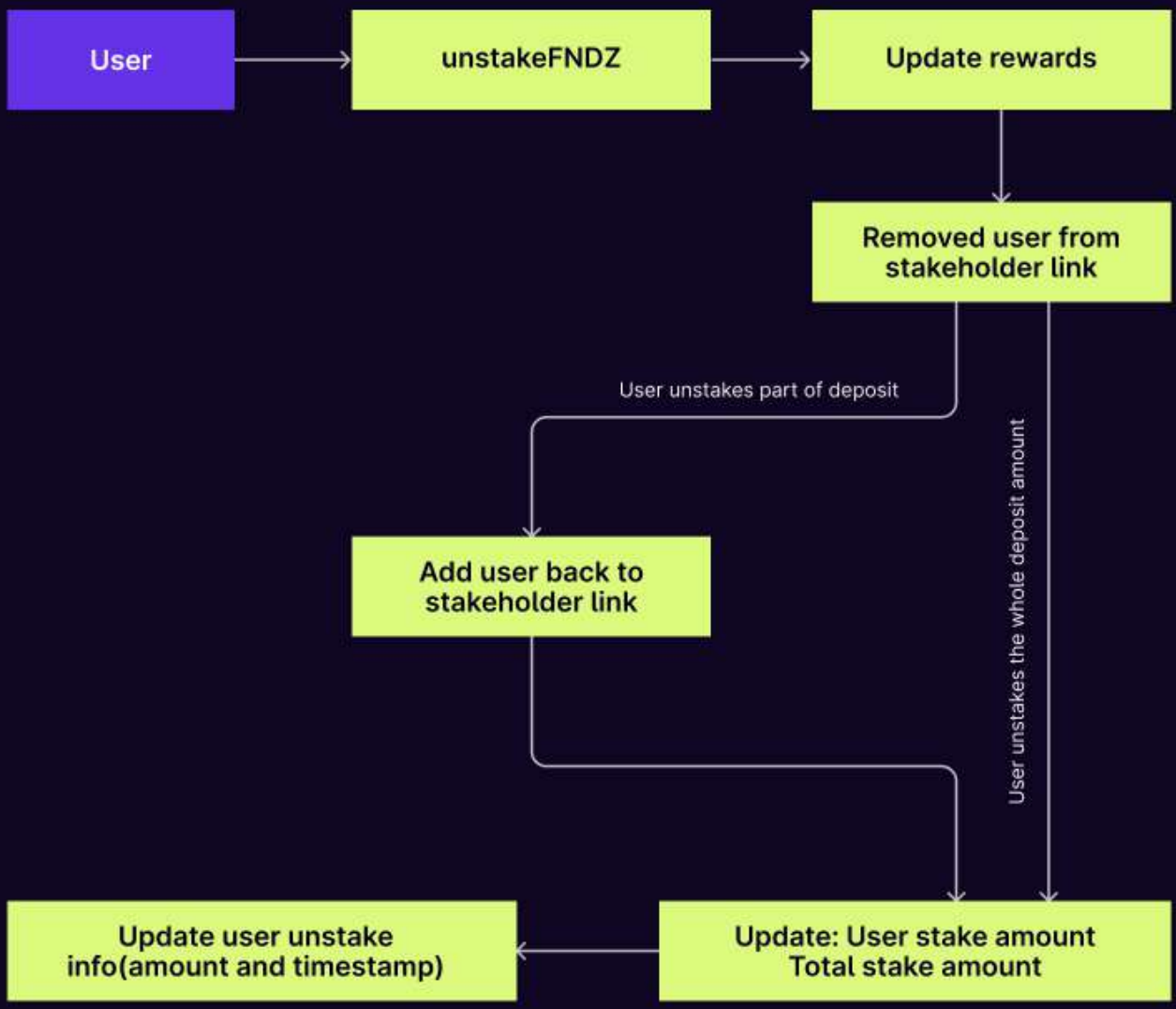


2. CLAIM REWARDS



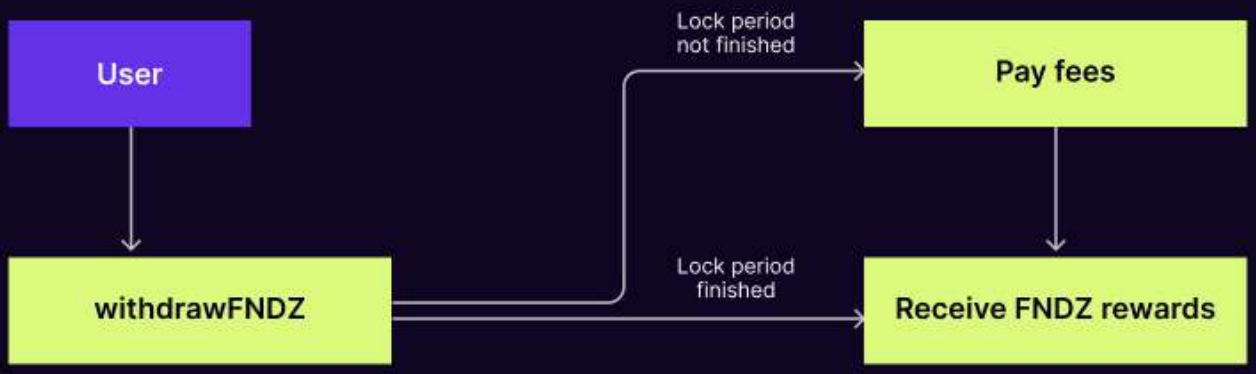
3. UNSTAKE FNDZ

This function marks that user wants to withdraw some amount of staked FNDZ tokens. Tokens then can withdrawn after lock period or before lock period. In second case, user will have to pay fees from withdrawn amount.



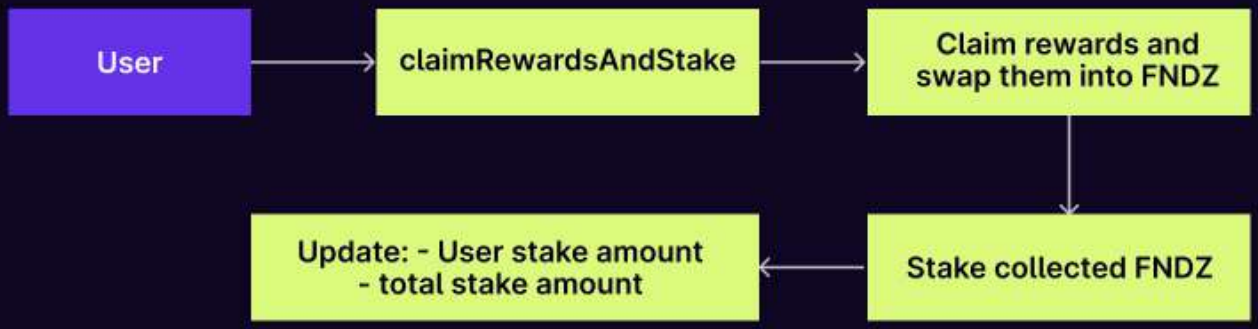
4. WITHDRAW FNDZ

Allows user to withdraw unstaked amount. In order not to pay fee from unstaked amount, user has to wait lock period.



5. CLAIM REWARDS AND STAKE FNDZ

Allows user to claim his rewards, swap them into FNDZ token and stake collected FNDZ.



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

MEDIUM | RESOLVED

Potential reentrancy attack to claim rewards multiple times.
FNDZStaking.sol: function __claimRewardsAsFNDZ().
Function performs swap of rewards to FNDZ token with swap routes which are passed by user. Since, anyone can create a pair on Uniswap factory, a path with malicious token might be passed, which is why all all storage variables should be changed before external calls.Currently, stakeHolder info "stakeUpdatedAt" and "hasEarnedRewards" are updated after the call. In case, a malicous token is used in a swap route with reentrancy to claiming rewards, "stakeUpdatedAt" should be already updated, so that user doesn't claim rewards multiple times.

Recommendation:
Update stakeHolderInfo (Currently on lines 483, 484) before the loop (Before line 462).

MEDIUM | RESOLVED

Function only returns 'true'.
ReferralRegistry.sol: function __isReferredAddress().
FNDZController.sol: isDenominationAssetApproved().
Function returns only 'true' value and not in all cases. Function should return values every time.

Recommendation:
Make sure to return value every time.

LOW | RESOLVED

Reward tokens without a direct pair with FNDZ in swap protocol cannot be withdrawn as FNDZ.

InlineSwapMixin.sol: function __inlineSwapAsset().
Function __inlineSwapAsset() is used by FNDZStaking for swapping reward tokens to FNDZ(FNDZStaking.sol: line 473). Function creates a swap route with direct pair only(Reward token → FNDZ), however, it is possible that there won't be a direct pair and swap will revert, canceling the withdrawal of reward in FNDZ. Issue is marked as low, because it is still possible to withdraw rewards directly without swapping them into FNDZ, or only reward tokens with direct swap pair might be used.

Recommendation:
One of the solutions would be to allow users to path swap routes. Another solution is to store swap routes for tokens in storage and create a setter for admin to set optimal swap routes in advance.

Post-audit:
It is possible to path swap routes to function now. This way, tokens without direct pair with FNDZ can be swapped.

LOW | RESOLVED

Iteration through array can consume great amount of gas.

ReferralRegistry.sol: migrateReferral().
FNDZController.sol: getIndex().
Iteration through array can consume a lot of gas. In case array has too many elements, iteration can consume more gas,
than can fit in a single block.

Recommendation:
Verify that function execution is optimized, and won't revert due to 'out of gas' error.

INFORMATIONAL | RESOLVED

Wrong index is verified.
FNDZStaking.sol: function removeTrackedAssets(), line 120.
Statement verifies that the found positional index of token to remove is the last. Currently a variable “i” is verified, which is a positional index of array “_assets”, not “trackedAssets”. A variable “j” should be verified here instead of “i”.

Recommendation:
Verify variable “j” instead of “i”.

INFORMATIONAL | RESOLVED

Function is not restricted.
FNDZController.sol: createNewFund().
FundDeployer.sol: createMigratedFundConfig(), cancelMigration(), cancelMigrationEmergency(), executeMigration(), executeMigrationEmergency().

Recommendation:
Verify that functions should not be restricted.

INFORMATIONAL | RESOLVED

ToDo in code
Dispatcher.sol: cancelMigration().
Pre-production contracts should not contain unfinished logic.

Recommendation:
Verify to implement all the logic.

Outdated Solidity version.
Currently the protocol utilizes Solidity version 0.6.12. It works well, though the general auditors security checklist recommends to utilize the newest stable version of Solidity which is 0.8.11. The newest version of Solidity includes last optimization for the compiler, last bugfixes and last features such as built-in overflow and underflow validations.

Recommendation:
Update Solidity version to the latest version 0.8.11.

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

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

	VaultLibBaseCore.sol	Dispatcher.sol	VaultLibBase1.sol
Re-entrancy	Pass	Pass	Pass
Access Management Hierarchy	Pass	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass	Pass
Unexpected Ether	Pass	Pass	Pass
Delegatecall	Pass	Pass	Pass
Default Public Visibility	Pass	Pass	Pass
Hidden Malicious Code	Pass	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass	Pass
External Contract Referencing	Pass	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass	Pass
Unchecked CALL Return Values	Pass	Pass	Pass
Race Conditions / Front Running	Pass	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass	Pass
Floating Points and Precision	Pass	Pass	Pass
Tx.Origin Authentication	Pass	Pass	Pass
Signatures Replay	Pass	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass	Pass

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

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

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

	PerformanceFee.sol	IntegrationManager.sol	ParaSwapV5Adapter.sol
Re-entrancy	Pass	Pass	Pass
Access Management Hierarchy	Pass	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass	Pass
Unexpected Ether	Pass	Pass	Pass
Delegatecall	Pass	Pass	Pass
Default Public Visibility	Pass	Pass	Pass
Hidden Malicious Code	Pass	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass	Pass
External Contract Referencing	Pass	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass	Pass
Unchecked CALL Return Values	Pass	Pass	Pass
Race Conditions / Front Running	Pass	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass	Pass
Floating Points and Precision	Pass	Pass	Pass
Tx.Origin Authentication	Pass	Pass	Pass
Signatures Replay	Pass	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass	Pass

	ParaSwapV5Actions Mixin.sol	PolicyManager.sol	InvestorWhitelist.sol
Re-entrancy	Pass	Pass	Pass
Access Management Hierarchy	Pass	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass	Pass
Unexpected Ether	Pass	Pass	Pass
Delegatecall	Pass	Pass	Pass
Default Public Visibility	Pass	Pass	Pass
Hidden Malicious Code	Pass	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass	Pass
External Contract Referencing	Pass	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass	Pass
Unchecked CALL Return Values	Pass	Pass	Pass
Race Conditions / Front Running	Pass	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass	Pass
Floating Points and Precision	Pass	Pass	Pass
Tx.Origin Authentication	Pass	Pass	Pass
Signatures Replay	Pass	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass	Pass

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

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

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

	FundActionsWrapper.sol	alueInterpreter.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

	InlineSwapMixin.sol	StakeOrderLinking.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

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by FNDZ team

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

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

ChainlinkPriceFeed Test Cases

- ✓ should be able to set and get a Stale Rate Threshold and verify the event (61ms)
- ✓ Should not be able to set stale rate threshold twice (361ms)
- ✓ does not allow the empty primitives to add with the given aggregator and rateAsset values
- ✓ does not allow the primitives and aggregators array length which are unequal (50ms)
- ✓ does not allow the primitives and rateAsset array length which are unequal (56ms)
- ✓ does not allow the aggregtor to be zero account (68ms)
- ✓ Should adds a list of primitives with the given aggregator and rateAsset values and emits the add event (64ms)
- ✓ Should get a list of primitives with the given aggregator and rateAsset values (88ms)
- ✓ does not allow the primitives and aggregators array length which are unequal while updating (44ms)
- ✓ does not allow the empty primitives to update with the given aggregator
- ✓ Should not be able to update primitive's aggregator which is not yet added (45ms)
- ✓ does not allow the same aggregtor to be added again while updating (105ms)
- ✓ Should update a primitives with the given aggregator and verify the update event (163ms)
- ✓ Should be able to remove a primitive and verify the emit event (107ms)
- ✓ _primitives args should not be empty while removing
- ✓ Should not be able to remove the primitive which is not added yet (47ms)
- ✓ Only FNDZController owner can add/remove/update primitives (167ms)
- ✓ Should be traded only if the token is added as a primitive by a vault (1389ms)
- ✓ Should get the ethUsdAggregator, WETH Token, rateAsset , unit variable value (202ms)
- ✓ Should return the correct FNDZController address

ComptrollerLib Tests

- ✓ Should match Default Denomination Asset with AddressZero
- ✓ Should match VaultProxy with AddressZero
- ✓ Cannot call the buyShares because the vault is not active (71ms)
- ✓ functions with onlyActive modifier cannot be called if vault is not Active (105ms)

Comptroller Proxy Test Suite

- ✓ functions with onlyFundDeployer modifier can't be call by vault Owner (72ms)
- ✓ functions with onlyOwner modifier can't be called by random user (81ms)
- ✓ Functions with onlyNotPaused modifier can't be called while paused (145ms)
- ✓ Can still withdraw from vaults when paused (286ms)
- ✓ Should not able to call buyShares again while the timelock span (984ms)
- ✓ Should not be able to redeem shares while within the timelock span (570ms)
- ✓ Should be able to redeem shares even in timelock span, but not be able to buy shares, when have the pending migration request (1304ms)
- ✓ Verify all CallOnExtension Actions (1764ms)
- ✓ can not directly call the permissionedVaultAction()
- ✓ Random users can call Gross value related methods (152ms)
- ✓ Able to get Lib Routes
- ✓ Override pause can be get and set by the owner (55ms)
- ✓ Able to get the shares action time lock
- ✓ Able to get the vault proxy using comptroller
- ✓ can not callOnExtension for an address other than fee manager or integration manager
- ✓ can not add the unapproved asset to the tracked assets (85ms)
- ✓ can calculate netShareValue using FundActionsWrapper (277ms)
- ✓ can get continuous fees using FundActionsWrapper (393ms)
- ✓ should able to call approveAssetSpender (468ms)

ComptrollerLib VaultCallOnContract Tests

- ✓ Should be able to call on vault (110ms)
- ✓ Should not be able to call unregistered calls (52ms)
- ✓ Only owner can call (39ms)
- ✓ Can not call when release status is paused (89ms)
- ✓ Should fail on invalid method calls (111ms)

AggregatedDerivativePriceFeed Test Suite

addDerivatives tests

- ✓ Should be able to add derivatives and price feeds (60ms)
- ✓ Only FundDeployer owner can call (46ms)
- ✓ Derivatives should not be empty (41ms)
- ✓ PriceFeeds and Derivatives array length should not be unequal
- ✓ Should not be able to add if already added
- ✓ Should not be able to use zero addresses (78ms)
- ✓ derivative should be an supported sythetix token (46ms)

State Getters tests

- ✓ Should be able to get the priceFeed of derivative
- ✓ Should be able to verify if an asset is a supported asset

updateDerivatives tests

- ✓ Should be able to update price feeds of the derivatives (61ms)
- ✓ Only FundDeployer owner can call
- ✓ Derivatives should not be empty

- ✓ PirceFeeds and Derivatives array length should not be unequal
- ✓ Should not be able to update if not added
- ✓ Should not be able to use zero addresses (43ms)
- ✓ Should not be able to update derivative with the same priceFeed again

removeDerivatives tests

- ✓ Should be able to update price feeds of the derivatives (45ms)
- ✓ Only FundDeployer owner can call (45ms)
- ✓ Derivatives should not be empty
- ✓ Should not be able to remove if not added

calcUnderlyingValues tests

- ✓ Should be able to calculate underlying values of a derivative (60ms)
- ✓ can not use the derivatives which is not added

Dispatcher Test Suite

constructor

- ✓ Should match address of dispatcher owner with deployer
- ✓ Should match dispatcher nominated owner with address Zero

setNominatedOwner

- ✓ can only be called by the contract owner (248ms)
- ✓ does not allow an empty next owner address
- ✓ does not allow the next owner to be the current owner
- ✓ does not allow the next owner to already be nominated (135ms)
- ✓ correctly handles nominating a new owner (125ms)

removeNominatedOwner

- ✓ can only be called by the contract owner (95ms)
- ✓ correctly handles removing the nomination (127ms)
- ✓ can only be called when there is a nominated owner

claimOwnership

- ✓ can only be called by the nominatedOwner (100ms)
- ✓ correctly handles transferring ownership (90ms)

setSharesTokenSymbol

- ✓ disallows a call by a random user
- ✓ correctly updates the SharesTokenSymbol

setCurrentFundDeployer

- ✓ disallows calling with account other than owner
- ✓ disallows empty address as nextFundDeployer
- ✓ does not allow _nextFundDeployer to be a non-contract (67ms)
- ✓ does not allow _nextFundDeployer to be currentFundDeployer (53ms)
- ✓ correctly handles valid fund deployer address update (256ms)

getFundDeployerForVaultProxy

- ✓ returns FundDeployer of a vaultProxy (423ms)

getMigrationTimelock

- ✓ returns the migrationTimeLock value

setMigrationTimelock

- ✓ Should be able to set the migrationTimelock
- ✓ Should not able to set the same migration time lock again (48ms)

FeeManager Test Suite

- ✓ Should be able to register fees and verify the event (193ms)
- ✓ Verify registerFees negative cases (293ms)
- ✓ Should be able to deregister fees and verify the event (66ms)
- ✓ Verify deregisterFees negative cases (208ms)
- ✓ Should be able to get registered Fees (40ms)
- ✓ Should be able to fetch enabled fees for fund (428ms)
- ✓ verify feeUsesGav on different fees (64ms)
- ✓ Verify setConfigForFund functionality (588ms)

FNDZController Test Suite

Denomination Assets

- ✓ Should not be able to add duplicate denomination assets
- ✓ Should be able to add denomination assets (82ms)
- ✓ Should able to remove denomination assets (166ms)
- ✓ Should not be able to remove denomination asset which is not approved
- ✓ Should not be possible to add or remove denomination assets if the caller is not the owner (57ms)
- ✓ Should be able to verify the approved denomination asset (58ms)

Fee Configurations

- ✓ Should be possible to get and set a new fee configuration (118ms)
- ✓ Should be possible to get and set an existing fee configuration (100ms)
- ✓ Should be possible to delete an existing fee configuration (87ms)
- ✓ Should be possible to set a fee configuration with zero parameters (53ms)
- ✓ Should not be possible to delete an uninitialized fee configuration (84ms)
- ✓ Should not be possible to set a fee configuration with mismatched array lengths (49ms)
- ✓ Should be able to set a fee configuration with many parameters (113ms)
- ✓ Should not be possible to set or delete a fee configuration if the caller is not the owner (94ms)

Creating new vaults

Fund Deployer Requirements

- ✓ does not allow an empty _fundOwner (52ms)
- ✓ does not allow an empty _denominationAsset (82ms)
- ✓ does not allow the release status to be Paused (77ms)

FNDZ Controller Requirements

- ✓ does not allow an unapproved denomination asset (59ms)
- ✓ Should not allow to create fund if fundDeployer is not set (56ms)
- ✓ Should create a new Fund (1551ms)
- ✓ Should not allow a fund to be created with an unregistered fee (70ms)
- ✓ Should not allow a fund to be created with a fee parameter value greater than the maximum (74ms)

- ✓ Should not allow a fund to be created with a fee parameter value smaller than the minimum (76ms)
- ✓ Should allow a fund to be created with fee parameter values equal to the minimum and maximum (491ms)
- ✓ Should allow a fund to be created with zero fee parameters (757ms)
- ✓ Should allow a fund to be created with multiple fees that have different parameter lengths (1234ms)
- ✓ Should update updatePerformanceFeeSplit and updateManagementFeeSplit based on the condition (201ms)

Updating state variables

- ✓ Should be able to update inline swap router (174ms)
- ✓ Should be able to update inline swap factory address (165ms)
- ✓ Should be able to update fndzStakingPool address (249ms)
- ✓ Should be able to update fndzDao address (258ms)
- ✓ Should be able to update fndzDaoDesiredToken address (272ms)
- ✓ Should be able to update inline swap allowances (208ms)
- ✓ Should be able to update paraSwapFee (126ms)

FNDZ Invest Fee Test Suite

- ✓ Should invest fndz token on deposit and update the locked amount (931ms)
- ✓ Should withdraw fndz token on redeem shares and update the locked amount (1501ms)
- ✓ Locked fndz tokens can not be traded (939ms)
- ✓ fndz tokens can be traded if bought through trading (2248ms)
- ✓ Migration should not affect the locked amount (2366ms)
- ✓ Payout of management fee should update the locked amount (2537ms)
- ✓ Payout of performance fee should update the locked amount (3084ms)
- ✓ Should not settle fee if investmentDue is 0 (443ms)
- ✓ Can able to get the Fee info of the Fund

FNDZ Invest Registry Test suite

- ✓ Only the owner can set the Fee address (319ms)
- ✓ Fee Address can not be zero address or the existing one (73ms)
- ✓ Only from the Fee contract alone can update the locked amount
- ✓ can read the locked amount of the vault

FNDZStaking test suite

Staking Tests

- ✓ should be able to stake FNDZ tokens (402ms)
- ✓ staking contract should be approved with the stake amount before staking (203ms)
- ✓ Should update the earned reward tokens if the stake is updated (556ms)
- ✓ Should update stake order link correctly (807ms)
- ✓ stake amount should be greater than zero

UnStake Tests

- ✓ Should be able to unstake total staked FNDZ tokens (404ms)
- ✓ Should be able to unstake staked FNDZ tokens partially (421ms)

- ✓ Can not unstake again if some tokens were already unstaked and not withdrawn (262ms)
- ✓ unstake amount should be lesser than or equal to staked amount (193ms)
- ✓ Should update the earned FNDZ tokens if the rewards applicable (482ms)
- ✓ Should update stake order link correctly (2077ms)
- ✓ unstake amount should be greater than zero

Withdraw Tests

- ✓ Should be able to withdraw (Total Stake) without Fee if the FNDZ unstaked and locking period completed (310ms)
- ✓ withdraw amount should be equal to unstaked amount (268ms)
- ✓ Fee should be settled when tokens not unstaked or unstaked but the locking period is not completed (688ms)
- ✓ Should update the earned FNDZ tokens before withdrawal (if not unstaked) (2580ms)
- ✓ withdrawal amount should be lesser than or equal to staked amount (if not unstaked) (162ms)
- ✓ Should update stake order link correctly (if not unstaked) (1556ms)
- ✓ withdraw amount should be greater than zero

Claim Rewards Tests

- ✓ Should be able to claim the applicable rewards as FNDZ Tokens (565ms)
- ✓ Should be able to claim the applicable rewards as Underlying Tracked Tokens (536ms)
- ✓ Should be able to claim and stake the claimed FNDZ tokens (538ms)
- ✓ Can not initiate claim if no rewards were applicable (336ms)
- ✓ Should provide the complete rewards if all the stakers has staked for an year then claimed and withdrawn the FNDZ (1897ms)
- ✓ Reward calculation verification with example scenario (2475ms)
- ✓ Claim rewards when pool has maximum no. of tracked assets (100914ms)
- ✓ Should be able to claim the earned rewards after withdrawing all the stakes (562ms)
- ✓ Should be able to claim the earned rewards after unstaking all the stakes (477ms)
- ✓ Should revert if invalid reward type is passed
- ✓ Should be able to skip assets on claim (1650ms)
- ✓ Should not change the rewards amount even the user unstake first and then claim (439ms)

State variables & calculation Tests

- ✓ getRewardBalance should return only the rewardable tracked asset balance (640ms)
- ✓ Should allocate the earned rewards properly (1224ms)
- ✓ Should be able to get the current rewards applicable for the user (519ms)
- ✓ Should be able to add tracked assets (64ms)
- ✓ Only the owner can add the tracked assets
- ✓ can not add duplicates (67ms)
- ✓ Should be able to remove the added tracked assets (108ms)
- ✓ Only the owner can remove the tracked assets
- ✓ can not remove the asset which is not in the tracked assets
- ✓ can get the array of tracked assets
- ✓ owner can update the unstakeFeeRate (68ms)
- ✓ owner can update the unstakeTimelock (76ms)

Fund Deployer

- ✓ Should return the new instance of FundDeployer once it's deployed
- ✓ Should have the release status to prelaunch
- ✓ Should verify for the valid VaultLib address
- ✓ Does not allow createNewFund to be called directly
- ✓ Should be able to read state variables (50ms)

Vault Call Registering Tests

Registering Vault Calls Tests

- ✓ Should be able to register vault calls (45ms)
- ✓ Should be able to verify registered vault calls (94ms)
- ✓ contracts list can not be empty (50ms)
- ✓ contracts and selectors length should be same (47ms)
- ✓ Should not be able to register a same call again (97ms)
- ✓ Only owner can call

Deregistering Vault Calls Tests

- ✓ Should be able to deregister vault calls (91ms)
- ✓ contracts list can not be empty (42ms)
- ✓ contracts and selectors length should be same (39ms)
- ✓ Should not be able to deregister a call which is not registered (56ms)
- ✓ Only owner can call

Integration Manager Test Cases

- ✓ able to add authenticated user for a fund (55ms)
- ✓ able to remove authenticated user of a fund (65ms)
- ✓ isAuthUserForFund should return false for unauthenticated users
- ✓ Fund Owner should be an Authenticated user
- ✓ Should not be able to add/remove fund owner (77ms)
- ✓ Only the fund owner can add/remove Authenticated user (51ms)
- ✓ Only activated fund's comptroller proxy can be used
- ✓ An account can be added only once (77ms)
- ✓ Should not be able to remove unauthenticated user
- ✓ Should emit the events (76ms)

Investor Whitelist Test Cases

- ✓ should return the identifier
- ✓ should return if an address passes the policy rule (174ms)
- ✓ should not allow addFundSettings to be called directly (79ms)
- ✓ should not allow updateFundSettings to be called directly (174ms)
- ✓ should not allow an unauthorized user to call enablePolicyForFund for a vault that they do not own (111ms)
- ✓ should allow an address to be added and removed in the same call (180ms)
- ✓ should not allow an already whitelisted address to be added again (367ms)
- ✓ should not allow an address to be removed if it is not already whitelisted (83ms)
- ✓ should enable and update a whitelist policy with add and remove the address in the whitelist (278ms)

- ✓ should emit AddressesAdded and AddressesRemoved events (145ms)
- ✓ should prevent non-whitelisted addresses from calling buyShares (2488ms)
- ✓ Should be able to get the list of whitelisted addresses (129ms)

Management Fee Suite (Vault Setup)

- ✓ Sets the scaledPerSecond rate correctly (398ms)
- ✓ Does not allow a scaledPerSecondRate of 0 (201ms)

Management Fee Suite (Vault Denomination Asset == FNDZ DAO Desired Currency)

- ✓ Does not trigger management fee during first shares purchase (52ms)
- ✓ Should be able to get the fee info
- ✓ Triggers management fee of 1% when invoking continuous hook after one year (655ms)
- ✓ Triggers management fee of 1% when buying shares after one year (831ms)
- ✓ Should transfer shares to fndzStaking and fndzDao instead of assets if the transfer failed (809ms)

Management Fee Suite (Vault Denomination Asset != FNDZ DAO Desired Currency)

- ✓ Triggers management fee of 1% when invoking continuous hook after one year (947ms)
- ✓ Triggers management fee of 1% when buying shares after one year (888ms)

MinMaxInvestment Tests

- ✓ sets state vars
- ✓ should return the identifier
- ✓ addFundSettings can only be called by the PolicyManager
- ✓ does not allow minInvestmentAmount to be greater than or equal to maxInvestmentAmount unless maxInvestmentAmount is 0 (142ms)
- ✓ updateFundSettings can only be called by the policy manager (139ms)
- ✓ returns false if the investmentAmount is out of bounds (499ms)
- ✓ Should able to create new fund with MaxInvestmentAmount 0 (352ms)
- ✓ can create a new fund with this policy, and it can disable and re-enable the policy for that fund (643ms)
- ✓ Vault test with min investment of 1 and a max of 2 (1954ms)
- ✓ Vault test with min investment of 0 and a max of 2 (1789ms)
- ✓ Vault test with min investment of 1 and a max of 0 (1760ms)

ParaSwapV5Adapter Test Cases

- ✓ Get the Identifier
- ✓ Should parse the encoded data correctly (70ms)
- ✓ Should not accept invalid selector signature
- ✓ Revert if encoded data is not valid
- ✓ Only Integration Manger can call the takeOrder method
- ✓ Get AugustusSwapper Address
- ✓ Get TokenTransferProxy Address

Performance Fee Test Cases (Vault Setup)

- ✓ Should be able to create a vault with Performance fee (10%) and fetch fee info (1328ms)
- ✓ Revert if Fee percentage is not within 0% - 30% (296ms)
- ✓ Revert if Crystallization period is not within minimum of one week to maximum of quarterly (111ms)

Performance Fee Test Cases (FNDZ DAO Desired Currency == Denomination Asset)

- ✓ Performance fee should be updated (mint/burn) on next deposit/withdraw after traded (2991ms)
- ✓ Performance Should not be updated if totalSharesSupply = 0 or totalSharesSupply == sharesOutstanding (2388ms)
- ✓ Outstanding Shares should not be paid out until the Crystallization period ends (1955ms)
- ✓ Payout should redeem all the outstanding shares, including unpaid shares from a previous period (3162ms)
- ✓ Payout should redeem all the outstanding shares, including newly minted shares after the previous period (2729ms)
- ✓ Payout should redeem all the outstanding shares, including the burned shares after the previous period (2701ms)
- ✓ Payout should split the performance fee to the owner, staking pool, and dao (1750ms)
- ✓ Withdrawing shares before the crystallization period won't affect performance obtained (1946ms)
- ✓ Verifying the high water mark works as expected (5318ms)
- ✓ Migrate should trigger the performance update before destruct (2946ms)
- ✓ Should transfer shares to fndzStaking and fndzDao instead of assets if the transfer failed (1508ms)

Performance Fee Test Cases (FNDZ DAO Desired Currency != Denomination Asset)

- ✓ Performance Fee splitted correctly and FndzDao only receives the desired token (1525ms)

Performance Fee Test Suite (Verify splits on Different Stakes)

- ✓ Shares Outstanding paid to Vault Owner should scale according to how much they have staked (37278ms)

PolicyManager Test Cases

- ✓ Should be able to add, get and remove policies (175ms)
- ✓ Should return true/false based on registered policies (179ms)
- ✓ Should emit register and deregister events (202ms)
- ✓ policies to register should not be empty (48ms)
- ✓ policies to deregister should not be empty
- ✓ Should not be able to register same policy multiple times (125ms)
- ✓ Should not be able to deregister a policy which is not registered (51ms)
- ✓ Only the FundDeployer Owner can register or deregister policies (64ms)
- ✓ Should be able to Enable and Get enabled Policies for a Vault (187ms)
- ✓ Should be able to check whether the policy is enabled or not (222ms)
- ✓ Only the fund owner can Enable/Update/Disable policies (281ms)
- ✓ Should not be able to enable a policy twice (205ms)
- ✓ Should not be able to use a policy which is not registered (40ms)
- ✓ Should be able to Update a Policy settings for a Vault (243ms)
- ✓ Should not be able to update a policy which is not enabled for the fund
- ✓ Should be able to Disable a Policy for a Vault (334ms)
- ✓ Should not be able to disable a policy which is not enabled for the fund
- ✓ Should emit enable and disable policy for fund event (212ms)

ChainLinkPriceAggregator Test Suite

- ✓ Oracle can update the Price of the tokens (62ms)
- ✓ Only accessor can update price feeds

- ✓ timestamp should only be the present one (190ms)
- ✓ _tokens cant have the zero address (40ms)
- ✓ _prices should be greater than zero (44ms)
- ✓ Should be able to update the oracle
- ✓ only owner can update the oracle (42ms)
- ✓ Should be able to fetch the price and timestamp (83ms)

PriceAggregatorProxy Test Suite

- ✓ Should be able read price and latest timestamp from aggregator (112ms)

ReferralRegistry test suite

- ✓ Buy Shares with referrer address should be added to ReferralRegistry (524ms)
- ✓ If referral exists, configured fee should be deducted and added to the referrer (1023ms)
- ✓ Referral fee should not be deducted if the buyer not referred (1099ms)
- ✓ Migration should not affect the added referrals (3545ms)
- ✓ Should be able to add a referral and verify whether its referred and can get the referral (562ms)
- ✓ Should not be added to the referrals if _referrer is zeroAddress (416ms)
- ✓ Should not be able to change the referrer once referred (1005ms)
- ✓ Referrer and Referee should not be same (427ms)
- ✓ Should not be able to get the referral if not referred and isReferred should return false (46ms)
- ✓ Only the owner can set the Fee address (278ms)
- ✓ Fee Address can not be zero address or other addresses or the existing one (72ms)

ComptrollerLib buyShares Tests

- ✓ Should place order of buyShares (346ms)
- ✓ Should revert if investment amount is 0 (173ms)
- ✓ Should revert with Shares received < _minSharesQuantity (398ms)

ComptrollerLib redeemSharesDetailed Tests

- ✓ Should Redeem Shares (558ms)
- ✓ does not allow a _sharesQuantity greater than the redeemer balance (473ms)
- ✓ does not allow duplicate _assetsToSkip (393ms)
- ✓ does not allow duplicate _additionalAssets (371ms)
- ✓ does not allow share redemption when there are no payout assets (533ms)
- ✓ Should use additionalAssets to received untracked assets on the vault (905ms)
- ✓ Should be able to forfeit assets when calling redeemSharesDetailed by using assetsToSkip (918ms)
- ✓ Should remove non-denominational tracked assets when the vault balance drops to 0 during share redemption (949ms)
- ✓ Should send the proportional number of assets relative to the percentage of vault shares redeemed (564ms)

Re-Entrancy Tests

- ✓ buyShares (249ms)
- ✓ redeemSharesDetailed (1132ms)
- ✓ callOnExtension (998ms)

StakeOrderLinking Test Suite

addToStakeOrderLink()

- ✓ should add address in proper order (324ms)
- ✓ should not add the zero address
- ✓ should not add the address which is already present in the link (76ms)

removeFromStakeOrderLink()

- ✓ should remove the address from the link (545ms)
- ✓ should not remove the zero address
- ✓ should not remove the address which is not present in the

linkSynthetixPriceFeed Test Suite

addSynths Tests

- ✓ Should be able to add synths (57ms)
- ✓ Only FundDeployer owner can call
- ✓ synths should not be empty
- ✓ Should not be able to add if added (82ms)
- ✓ Revert if currencyKey of synth is 0 (77ms)

updateSynthCurrencyKeys Tests

- ✓ Should be able to update currencyKey of synths (77ms)
- ✓ synths should not be empty
- ✓ Should not be able to update if not added
- ✓ Should not be able to update using same currencyKey

calcUnderlyingValues tests

- ✓ Should be able to calculate underlying values of a derivative (45ms)
- ✓ Revert if synth is not added
- ✓ Revert if invalid Rate (82ms)

State getter tests

- ✓ Should be able to check supported synths
- ✓ can get the Address Resolver address
- ✓ can get the list of currencyKeys of synths
- ✓ can get the currencyKey of a synth
- ✓ can get the sUSD address

Paraswap Trade

- ✓ Should Make Trade (497ms)
- ✓ Should Make Trade and Deduct Partner Fee If partnerAddress and partnerFee present (652ms)
- ✓ Authorized users only allowed to create trade (60ms)
- ✓ Adaptor Should be Registered (97ms)
- ✓ Incoming asset address of Encode Data Should be Valid (154ms)
- ✓ Minimum Incoming asset of Encode Data Should be greater than zero (148ms)
- ✓ Insufficient Balance to Trade (316ms)
- ✓ Incoming asset should be present in prmitive or derivative assets list (199ms)
- ✓ outgoing asset should not be invalid (169ms)
- ✓ outgoing asset amount should be greater than zero (182ms)
- ✓ Trade does not allow invalid _extension (42ms)
- ✓ Trade does not allow invalid _actionId (60ms)

- ✓ Revert if _callArgs is not valid (66ms)
- ✓ Revert if received amount less than expected (661ms)
- ✓ Should be able to trade from air-dropped unapproved asset to approved asset (643ms)

Upgradable Contract Test Suite

- ✓ Should be able to upgrade FNDZController (1000ms)
- ✓ Should be able to upgrade ReferralRegistry (1445ms)

Value Interpreter Tests

- ✓ Should revert with Arrays unequal lengths (247ms)
- ✓ Should revert with Unsupported _quoteAsset (167ms)
- ✓ Should verify primitive price feed
- ✓ Should verify Aggregated Derivative Price Feed
- ✓ Should calculate total asset value (79ms)
- ✓ Should calculate total asset value (109ms)
- ✓ Should be able to calculate live assets total value using calcLiveAssetsTotalValue (98ms)

calcCanonicalAssetValue for Derivatives

- ✓ Should be able to calculate asset value of derivative (453ms)
- ✓ Should revert if the _baseAsset is not added in derivatives (56ms)
- ✓ Should revert if the _quoteAsset is not a supported primitve

VaultLib Tests

- ✓ Should match accessorValue with AddressZero
- ✓ Should match Creator Value with AddressZero
- ✓ Should match Migrator Value with AddressZero
- ✓ Should match ownerValue with AddressZero
- ✓ Should match trackedAssetValue with AddressZero
- ✓ Vaultlib name should be empty
- ✓ Should match Decimal Value To 18

Vault Migration Test Suite

- ✓ Should be able to migrate vault on new release (2663ms)
- ✓ Should be able to create Migrated fund config only in live release (71ms)
- ✓ Should be able to signal the migration only in live release (310ms)
- ✓ Only the new comptroller proxy creator can signal the migration (369ms)
- ✓ The migrator cannot execute a rogue migration, and can ignore the rogue migration (481ms)
- ✓ The migrator cannot replace the comptroller address with an arbitrary smart contract when they signal a migration (335ms)
- ✓ Only current fund deployer can signal the migration (236ms)
- ✓ Only the permissioned migrator can signal the migration (378ms)
- ✓ signal migration is possible only on new fund deployer (213ms)
- ✓ Should not allow to execute migration without signaling (52ms)
- ✓ Only the newFund deployer can execute the migration (613ms)
- ✓ Only the permissioned migrator can execute the migration (871ms)
- ✓ Only the current fund deployer can execute the pending migration (387ms)
- ✓ Should be able to cancel the migration request (529ms)

- ✓ Can not cancel the migration if the migration not signaled (113ms)
- ✓ Can not cancel the migration if the releaseStatus is not Live (245ms)
- ✓ Only the new fund deployer or migrator can cancel the migrations (527ms)

Migration with bypass enabled

- ✓ Should be able to migrate with the bybass failure enabled (1407ms)
- ✓ Should be able to cancel the migration request with bybass failure enabled (2096ms)

Tests on setMigrator of vault

- ✓ Should be able to set migrator of the vault
- ✓ only the owner can set the migrator of the vault
- ✓ cant set the same migrator twice (66ms)

Tests on migration related Dispatcher methods

getTimelockRemainingForMigrationRequest

- ✓ returns remaining time lock for migration else zero (174ms)

hasMigrationRequest

- ✓ returns false if vault does not have any migration request
- ✓ returns true if vault has the migration request (100ms)

hasExecutableMigrationRequest

- ✓ returns false if vault does not have executable migration request (146ms)
- ✓ returns true if vault has the executable migration request (128ms)

cancelMigration

- ✓ Only the permissioned migrator can call (122ms)

VaultProxy tests

- ✓ Should match vault proxy creater with dispatcher
- ✓ Should match migrator value with AddressZero
- ✓ Should match vault owner value with Fund Owner Address
- ✓ Should match vault proxy name with fund name
- ✓ Should match vault proxy symbol with fund Denomination asset (39ms)
- ✓ Should match vault proxy decimal length to 18
- ✓ Should revert approve with Unimplemented (253ms)
- ✓ Should revert transfer with Unimplemented (97ms)
- ✓ Should revert transferFrom with Unimplemented (166ms)
- ✓ Add Tracked Asset cannot be called by an unauthorized user on the fund (92ms)
- ✓ Should allow an authorized user on the fund to add a zero balance tracked asset (159ms)
- ✓ Should allow an authorized user on the fund to add a non-zero balance tracked asset (332ms)
- ✓ Should not able to remove non-zero balance tracked asset (619ms)
- ✓ remove tracked Asset (415ms)
- ✓ Removed Tracked Asset cannot be called by an unauthorized user on the fund (107ms)
- ✓ Withdraw Asset To can only be called by the accessor (93ms)
- ✓ callOnContract cannot be called directly (48ms)
- ✓ approveAssetSpender cannot be called directly
- ✓ redeemVirtualShares cannot be called directly
- ✓ redeemAndSwapVirtualShares cannot be called directly

```
✓ burnShares cannot be called directly (38ms)
✓ mintShares cannot be called directly
✓ transferShares cannot be called directly (92ms)
✓ swapAndWithdrawAssetTo cannot be called directly (323ms)
✓ addTrackedAsset cannot be called directly
✓ removeTrackedAsset cannot be called directly
✓ swapAsset is only accessible by the respective comptroller
437 passing (45m)
```

FILE	% STMTS	% BRANCH	% FUNCS
ChainlinkPriceAggregator.sol	100	90	100
FNDZController.sol	100	96.88	100
FNDZInvestmentRegistry.sol	100	100	100
FNDZStaking.sol	98.65	89.66	100
PriceAggregatorProxy.sol	100	100	100
ReferralRegistry.sol	100	81.25	100
Dispatcher.sol	99.12	92.42	100
VaultLibBase1.sol	100	100	100
VaultLibBaseCore.sol	100	50	100
VaultProxy.sol	100	50	100
FundDeployer.sol	98.77	81.25	100
ComptrollerLib.sol	96.33	81.82	98.18
ComptrollerProxy.sol	100	100	100
VaultLib.sol	94.67	84.93	96
FeeManager.sol	96.15	82.89	100
EntranceReferralFee.sol	96.15	83.33	85.71
FNDZInvestmentFee.sol	100	83.33	100
ManagementFee.sol	87.5	64.29	100

PerformanceFee.sol	98.21	86.36	100
IntegrationManager.sol	86.08	74.49	83.33
ParaSwapV5Adapter.sol	100	100	100
ParaSwapV5ActionsMixin.sol	100	100	100
PolicyManager.sol	100	89.29	100
InvestorWhitelist.sol	100	100	100
MinMaxInvestment.sol	94.44	100	87.5
PreBuySharesValidatePolicy Base.sol	100	100	100
AddressListPolicyMixin.sol	100	75	100
PolicyBase.sol	80	100	80
ExtensionBase.sol	63.64	50	33.33
FundDeployerOwnerMixin.sol	75	100	75
PermissionedVaultAction Mixin.sol	100	100	100
AggregatedDerivativePrice Feed.sol	100	100	100
SynthetixPriceFeed.sol	100	100	100
ValueInterpreter.sol	100	90.91	100
FundActionsWrapper.sol	100	100	100
InlineSwapMixin.sol	91.3	50	100
StakeOrderLinking.sol	100	100	100
All files	**93.51**	**81.26**	**92.37**

Zokyo Secured team has carefully checked the whole set of unit tests and checked the original coverage of those tests. During the audit tests were verified for the correctness, wholesomeness, sufficient coverage, meaning of the scenarios, overall structure and the correct implementation (in spite of the covered functionality).

As the result - all tests are verified and the coverage is evaluated as conforming for security requirements. Original functionality has necessary coverage, standard and sub-standard functionality was veried by Zokyo Secured auditors in separate set of exploratory testing scenarios.

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Security team

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

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

```
FNDZStaking
    ✓ Should stake FNDZ (320ms)
    ✓ Should claim rewards (178ms)
    ✓ Should unstake staked FNDZ (167ms)
    ✓ Should withdraw FNDZ after unstake time lock (147ms)
    ✓ Crash if uniswapV2Pair does not exist (556ms)
  5 passing (27m)
```

Notice! Zokyo Secured team has checked the whole set of contracts to operate correctly, especially with AMM protocols. Additional set of unit-tests were written to make sure, that staking contract works correctly, as well as cover some uncovered cases.

As the result - all tests are verified, the flows of users' funds are verified not to get stucked at any stage of protocol operation. Original functionality is well-tested, reviewed by Zokyo Secured team and, as a conclusion, has all necessary security checks.

We are grateful to have been given the opportunity to work with the FNDZ 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 FNDZ team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo