SMART CONTRACT AUDIT

ZOKYO.

Feb 10, 2021 | v. 2.0

PASS

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

TECHNICAL SUMMARY

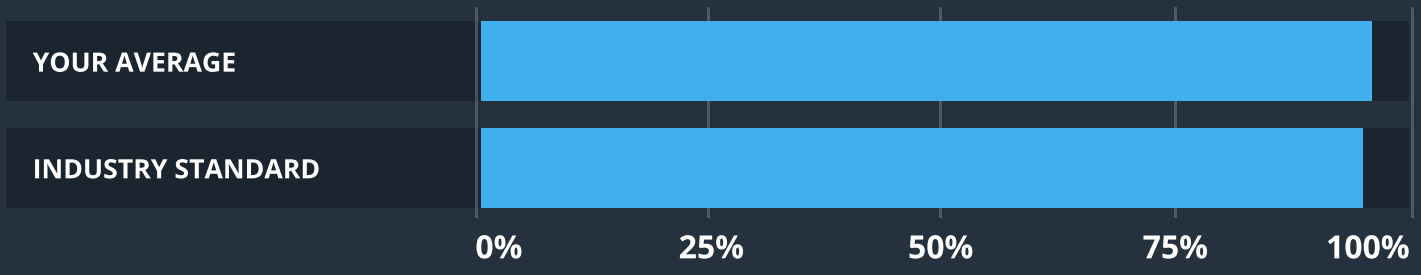
This document outlines the overall security of the Nord Finance smart contracts, evaluated by Zokyo's Blockchain Security team.

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

Contract Status

There was 1 critical issue found during the audit which was successfully resolved.

Testable Code



Testable code is 95.99 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 Nord Finance 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 following github repository – https://github.com/nordfinance/nordsavings-v1.
Commit id – 101c28276ad7702d1a6c979db10ea8c7fb1179f7.

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

- Implements and adheres to existing Token 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 gas, 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 Nord Finance smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1	Due diligence in assessing the overall code quality of the codebase.	3	Testing contract logic against common and uncommon attack vectors.
2	Cross-comparison with other, similar smart contracts by industry leaders.	4	Thorough, manual review of the codebase, line-by-line.

EXECUTIVE SUMMARY

Zokyo's Security team has conducted an audit of the contract with a detailed review of the codebase and testing the vulnerabilities.

We can admit the code is well written and has an appropriate style.

The rest of our findings mainly refer to optimizations and Solidity coding standards. Hence, the issues identified pose no threat to the safety of the contract deployment.

Zokyo's Security team would like to assess Nord Finance's smart contracts from two aspects: Technical and Economical.

Technical Summary. Based on the code analyzed, we can give a Good score to the codebase provided. The issues found and described below are stopping us from giving the Excellent score.

Economical Summary. Given the growing demand for review against possible economical exploits, we recommend every interested party to get acquainted with the findings below describing potential discrepancies, specific nature of how certain elements of the Nord Finance ecosystem.

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

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.

MANUAL REVIEW

CRITICAL | RESOLVED

Calling twice safeApprove or approve methods and passing first time 0 and second time amount of tokens does not solve potential double spending problem as they should be executed as separate transactions and after first transaction mined to block, a user has to verify that spender did not spend any tokens approved before.

Recommendation:
Reimplement mitigation of double spend problem or remove approval to 0 as it does not handle double spending problem.

HIGH | UNRESOLVED

Governance contract is not audited as the repository is not including it.

Recommendation:
Include reference to existing governance contract or add it to the repository.

HIGH | UNRESOLVED

Method withdraw of Vault contract is not triggering recalculation of SharePriceCheckpoint.

HIGH | UNRESOLVED

AAveStrategy & CompoundStategy methods (startSaving, investAllUnderlying, withdrawAllToVault, withdrawToVault) can be called by Controller or Governance bypassing Vault without recalculation of sharePriceCheckpoint.

HIGH | RESOLVED

RewardToken supply is minted to contract deployer.

Recommendation:
RewardToken total supply should belong to TokenPools utilized by TokenGeyser.

HIGH | RESOLVED

AaveStrategy, CompoundStrategy contracts are missing a predefined list of addresses that can be used to verify correct initialization of them as they specifically have to be used with specific contracts.

Recommendation:
Define allowed list of contracts addresses with reference to chain id
Comment: Smart contracts are pointed to mainnet smart contract addresses along with test framework is configured to utilize the last blockchain snapshot for testing.

HIGH | RESOLVED

Solidity files are not pointing to specific Solidity compiler version.

Recommendation:
Use specific Solidity compiler version.

MEDIUM | RESOLVED

package.json, development dependency (@babel/core, app-root-path, dotenv) added to dependencies.

Recommendation:
It is expected that development dependencies should be located in devDependency.

MEDIUM | RESOLVED

Vault contract has incorrect event names (dFeeDeposited, totalFeeCollected), it is expected that event names are declared in PascalCase.

MEDIUM | RESOLVED

DRY, slot manipulation methods (setBoolean, getBoolean, setAddress, getAddress, setUint256, getUint256) are duplicated across two contracts.

Recommendation:
Move those methods to separate contract or library.

MEDIUM | RESOLVED

Method finalizeUpgrade of Vault contract can be called multiple times, that can cause unexpected side effects.

MEDIUM | RESOLVED

Solidity files has no license declaration.

Recommendation:
Specify license in every Solidity file.

MEDIUM | RESOLVED

ComptrollerInterface is not reflecting updated state property `markets` output arguments.

Recommendation:
Add an extra bool output variable.

LOW | RESOLVED

Smart contracts AaveStrategy, FundDivisionStrategy, CompoundInteractor, CompoundStrategy, NoopStrategy are using the wrong naming convention for state variable underlying_erc.

Recommendation:
All state variables should be named using the camel-case naming convention.

LOW | RESOLVED

Method getATokenContract from contract AaveStrategy has incorrect comment referring to kovan network.

Recommendation:
Change comment to the proper one.

LOW | RESOLVED

Uniswap interface contracts should be added as NPM dependency instead of copying sources.

LOW | RESOLVED

ClaimRewardProxy.removeRewardDistributor, FundDivisionStrategy.unwhitelistStrategy methods are using nonoptimal removal approach.

Recommendation:
Reassign the last element to the position of the element to be deleted and call method pop on that array.

INFORMATIONAL | UNRESOLVED

Constructors at CompoundStrategy, VaultStorage, GovernableInit, AaveStrategy does extra check to verify that passed arguments to the constructor are equal to predefined values.

Recommendation:
Initialize Smart contracts state variables with corresponding predefined values instead of passing them inside of the constructor.

Vulnerabilities Checklist

	Vault	VaultProxy	TokenGeyser
Re-entrancy	Not affected	Not affected	Not affected
Access Management Hierarchy	Not affected	Not affected	Not affected
Arithmetic Over/Under Flows	Not affected	Not affected	Not affected
Unexpected Ether	Not affected	Not affected	Not affected
Delegatecall	Not affected	Not affected	Not affected
Default Public Visibility	Not affected	Not affected	Not affected
Hidden Malicious Code	Not affected	Not affected	Not affected
Entropy Illusion (Lack of Randomness)	Not affected	Not affected	Not affected
External Contract Referencing	Not affected	Not affected	Not affected
Short Address/ Parameter Attack	Not affected	Not affected	Not affected
Unchecked CALL Return Values	Not affected	Not affected	Not affected
Race Conditions / Front Running	Not affected	Not affected	Not affected
General Denial Of Service (DOS)	Not affected	Not affected	Not affected
Uninitialized Storage Pointers	Not affected	Not affected	Not affected
Floating Points and Precision	Not affected	Not affected	Not affected
Tx.Origin Authentication	Not affected	Not affected	Not affected

Signatures Replay	Not affected	Not affected	Not affected
Pool Asset Security (backdoors in the underlying ERC-20)	Not affected	Not affected	Not affected

	TokenPool	AaveStrategy
Re-entrancy	Not affected	Not affected
Access Management Hierarchy	Not affected	Not affected
Arithmetic Over/Under Flows	Not affected	Not affected
Unexpected Ether	Not affected	Not affected
Delegatecall	Not affected	Not affected
Default Public Visibility	Not affected	Not affected
Hidden Malicious Code	Not affected	Not affected
Entropy Illusion (Lack of Randomness)	Not affected	Not affected
External Contract Referencing	Not affected	Not affected
Short Address/ Parameter Attack	Not affected	Not affected
Unchecked CALL Return Values	Not affected	Not affected
Race Conditions / Front Running	Not affected	Not affected
General Denial Of Service (DOS)	Not affected	Not affected
Uninitialized Storage Pointers	Not affected	Not affected

Floating Points and Precision	Not affected	Not affected
Tx.Origin Authentication	Not affected	Not affected
Signatures Replay	Not affected	Not affected
Pool Asset Security (backdoors in the underlying ERC-20)	Not affected	Not affected

	CompoundStrategy	Controller
Re-entrancy	Not affected	Not affected
Access Management Hierarchy	Not affected	Not affected
Arithmetic Over/Under Flows	Not affected	Not affected
Unexpected Ether	Not affected	Not affected
Delegatecall	Not affected	Not affected
Default Public Visibility	Not affected	Not affected
Hidden Malicious Code	Not affected	Not affected
Entropy Illusion (Lack of Randomness)	Not affected	Not affected
External Contract Referencing	Not affected	Not affected
Short Address/ Parameter Attack	Not affected	Not affected
Unchecked CALL Return Values	Not affected	Not affected
Race Conditions / Front Running	Not affected	Not affected

General Denial Of Service (DOS)	Not affected	Not affected
Uninitialized Storage Pointers	Not affected	Not affected
Floating Points and Precision	Not affected	Not affected
Tx.Origin Authentication	Not affected	Not affected
Signatures Replay	Not affected	Not affected
Pool Asset Security (backdoors in the underlying ERC-20)	Not affected	Not affected

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Nord Finance team (original test coverage)

Contract: ClaimRewardProxy Test
- Claim Reward and GetRewardBalance
  - Passed
    - ✓ claim reward from all token distributor (2990ms)
  - Passed
    - ✓ add and remove distributors (4285ms)

Contract: FundDivisionStrategy Test
- FundDivisionStrategy
  - ✓ Investment ratios and caps are enforced (5999ms)
  - ✓ Making Vault with partial transfer to strategy (18365ms)
  - ✓ Announces whitelist of a new strategy (2149ms)
- Unwhitelisting
  - ✓ Unwhitelists an empty strategy at the beginning/middle of the list (268ms)
  - ✓ Unwhitelists an empty strategy at the end of the list (297ms)

Contract: Kovan Aave DAI Strategy
- Kovan Aave DAI earnings
  - ✓ A user investing underlying and withdraws all (42370ms)
  - ✓ A user investing underlying and withdraws partial (6088ms)

Contract: Controllable Test
- Controller setting
  - ✓ set and read (203ms)
  - ✓ set storage (164ms)

Contract: Controller Test
- Deposit and Withdraw
  - ✓ Controller can set reward collector (98ms)
  - ✓ Controller can add vault and strategy (1354ms)
  - ✓ can add strategy for vault (6466ms)
  - ✓ can update vault strategy from controller (5687ms)

✓ withdraw all from controller (4057ms)
✓ Governance can salvage (463ms)
✓ Governance can salvage strategy (475ms)

Contract: Kovan Compound DAI Strategy
Kovan Compound DAI earnings
✓ A user investing underlying and withdraws all (66436ms)
✓ A user investing underlying and withdraws partial (14427ms)
✓ User owns comp which is liquidated (14198ms)
✓ Emergency Liquidity Shortage (17801ms)

Contract: Governable Test
Governance setting
✓ set and read (168ms)
✓ set storage (201ms)

tokenPool
balance
✓ should return the balance of the token pool (540ms)
transfer
✓ should let the owner transfer funds out (424ms)
✓ should NOT let other users transfer funds out (336ms)
rescueFunds
✓ should let owner users claim excess funds completely (204ms)
✓ should let owner users claim excess funds partially (158ms)
✓ should NOT let owner claim more than available excess funds (73ms)
✓ should NOT let owner users claim held funds (121ms)
✓ should NOT let other users users claim excess funds (105ms)

Contract: Upgradibility Test
Deposit and Withdraw
✓ reverts (4743ms)
✓ deposit and withdraw test (6072ms)
✓ Strategy Update (6490ms)
Upgradability test
✓ Interaction with vault being upgraded (18186ms)

```
Contract: Storage Test
  Storage setting
    ✓ governance can update governance (216ms)

Contract: Vault Test
  Deposit and Withdraw
    ✓ empty vault (1201ms)
    ✓ reverts (5859ms)
    ✓ deposit and withdraw test with a token of 6 decimals (12943ms)
    ✓ deposit and withdraw test (3445ms)
    ✓ deposit for and withdraw test (4299ms)
    ✓ withdraw all to vault (3857ms)
    ✓ startSaving test (5266ms)
    ✓ Strategy Update (1481ms)
    ✓ setting investment ratio (3305ms)
  Upgradability test
    ✓ Interaction with vault being upgraded (18649ms)

47 passing (14m)
```

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts/	96.21	70.83	94.62	96.10	
Controllable.sol	100.00	50.00	100.00	100.00	
ControllableInit.sol	75.00	50.00	80.00	66.67	16, 20
Controller.sol	86.11	46.43	77.78	87.18	... 114, 118, 163
Governable.sol	100.00	100.00	100.00	100.00	
GovernableInit.sol	75.00	50.00	85.71	81.82	40, 41
RewardToken.sol	100.00	100.00	100.00	100.00	
Storage.sol	100.00	100.00	100.00	100.00	
Vault.sol	98.54	79.69	97.83	98.56	179, 353
VaultProxy.sol	100.00	75.00	100.00	100.00	
VaultStorage.sol	100.00	100.00	100.00	100.00	
contracts/Interfaces/	100.00	100.00	100.00	100.00	
IController.sol	100.00	100.00	100.00	100.00	
IMigrator.sol	100.00	100.00	100.00	100.00	
IStrategy.sol	100.00	100.00	100.00	100.00	
IUpgradeSource.sol	100.00	100.00	100.00	100.00	
IVault.sol	100.00	100.00	100.00	100.00	
contracts/TokenDistribution/	90.36	58.82	86.67	90.00	
ClaimRewardProxy.sol	100.00	66.67	100.00	100.00	
IStaking.sol	100.00	100.00	100.00	100.00	
TokenGeyser.sol	89.20	56.67	80.00	88.76	... 725, 759, 774
TokenPool.sol	100.00	100.00	100.00	100.00	
contracts/compound/	100.00	100.00	100.00	100.00	
CTokenInterfaces.sol	100.00	100.00	100.00	100.00	

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
ComptrollerInterface.sol	100.00	100.00	100.00	100.00	
InterestRateModel.sol	100.00	100.00	100.00	100.00	
contracts/strategies/	90.29	51.47	76.00	90.83	
FundDivisionStrategy.sol	90.29	51.47	76.00	90.83	... 315, 440, 444
contracts/strategies/aave/	76.32	33.33	87.50	80.56	
AToken.sol	100.00	100.00	100.00	100.00	
AaveStrategy.sol	76.32	33.33	87.50	80.56	... 207, 227, 231
LendingPool.sol	100.00	100.00	100.00	100.00	
contracts/strategies/compound/	78.13	42.11	82.76	79.17	
CompleteCToken.sol	100.00	100.00	100.00	100.00	
CompoundInteractor.sol	80.95	37.50	71.43	80.95	43, 54, 55, 76
CompoundStrategy.sol	77.33	43.33	86.36	78.67	... 225, 261, 265
All files	**90.11**	**58.33**	**89.57**	**90.59**	

Tests written by Zokyo Secured team

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

Tests were based on the functionality of the code, as well as a review of the Nord Finance contract to cover untested lines of code.

Contract: AaveStrategy Contract

AaveStrategy create phase

- ✓ should lending pool address be correctly (1579ms)
- ✓ should controller be correct (54ms)
- ✓ should governance be correct (59ms)
- ✓ should initial AToken balance be correct (5071ms)

Aave strategy deposit phase

- ✓ should invested underlying balance be correctly (8177ms)
- ✓ should AToken balance be correct (4429ms)
- ✓ shouldn't start saving with zero amount in strategy (974ms)

Aave strategy start withdraw

- ✓ should withdraw to vault correctly (4983ms)
- ✓ should withdraw all to vault correctly (4164ms)
- ✓ shouldn't salvage not salvagable token (496ms)

Contract: ClaimRewardProxy Contract

ClaimRewardProxy creating

- ✓ should initialize claim reward proxy correctly (272ms)
- ✓ should initialize claim reward proxy only once (306ms)
- ✓ shouldn't initialize claim reward proxy with zero number of active reward distributor (88ms)
- ✓ should add reward distributor correctly (6761ms)
- ✓ should remove reward distributor correctly (534ms)
- ✓ should claim aggregated rewards correctly (4462ms)

Contract: Controller Contract

Controller creating phase

- ✓ should set fee reward forwarder correctly (179ms)
- ✓ should add to grey list correctly (493ms)
- ✓ should remove from grey list correctly (588ms)

✓ should set rewards distributor for vault correctly (2183ms)
✓ shouldn't set fee reward forwarder with zero address (103ms)

Controller vault and strategy
✓ should add vault and strategy correctly (1382ms)
✓ shouldn't add vault and strategy with zero address of vault and strategy (331ms)
✓ shouldn't add already existed vault (1085ms)

Controller deposit phase
✓ should add saver correctly (308ms)
✓ should remove saver correctly (733ms)
✓ should deposit and withdraw correctly (5041ms)

Controller set strategy phase
✓ should set strategy correctly (2722ms)
✓ should salvage correctly (605ms)
✓ should salvage from strategy correctly (514ms)

Contract: CompoundStrategy Contract

CompoundStrategy setup
✓ should create correctly (4822ms)
✓ should controller be correct
✓ should governance be correct (51ms)
✓ should underlying be correct
✓ should set allow liquidity shortage correctly (175ms)
✓ should getLiquidity be correct (817ms)

CompoundStrategy deposit phase
✓ should deposit and withdraw correctly (12948ms)
✓ should withdraw to vault from strategy (1074ms)
✓ should emergency exit and withdraw correctly (35997ms)
✓ should salvage correctly (936ms)
✓ shouldn't salvage not salvagable token (454ms)
✓ should withdraw correctly when sellLPToken not allow (1659ms)

Contract: FundDivisionStrategy Contract

FundDivisionStrategy init
✓ should init splitter correctly (3993ms)
✓ shouldn't init already init splitter (2924ms)
✓ shouldn't underlying token and vault equal zero address (237ms)
✓ shouldn't set not matched in vault underlying token (342ms)
✓ should set configure strategies corectly (6005ms)

FundDivisionStrategy set strategy
- ✓ should announce strategy whitelist correctly (7517ms)
- ✓ shouldn't announce strategy whitelist when underlying of splitter doesn't match strategy underlying (5838ms)
- ✓ shouldn't announce strategy whitelist when splitter is incorrect (5062ms)
- ✓ should whitelist strategy correctly (8103ms)
- ✓ should unwhitelist strategy correctly (5842ms)
- ✓ shouldn't unwhitelist zero address strategy (5079ms)
- ✓ shouldn't unwhitelist not whitelisted strategy (5516ms)
- ✓ shouldn't unwhitelist not empty strategy (9123ms)

FundDivisionStrategy invest
- ✓ should invest into strategy (4577ms)

FundDivisionStrategy withdraw
- ✓ should withdraw to vault correctly (9364ms)
- ✓ shouldn't withdraw to vault when amount exeed balance (9373ms)
- ✓ should withdraw all to vault correctly (9152ms)
- ✓ should withdraw to vault with zero caps correctly (9223ms)
- ✓ should satrt saving correctly (7050ms)
- ✓ should salvage correctly (633ms)

Contract: Governable Contract

Governance creating
- ✓ should create correctly (54ms)
- ✓ should governance be correct

Governance set storage
- ✓ should set storage correctly (129ms)
- ✓ shouldn't set storage with zero address (61ms)
- ✓ shouldn't set storage by not a governance (58ms)

Contract: Storage Contract

Storage governance
- ✓ should set new governance by governance (63ms)
- ✓ shouldn't set new governance by not a governance (79ms)
- ✓ shouldn't new governance equal a zero address (57ms)

Storage controller
- ✓ should set controller by governance (74ms)
- ✓ shouldn't set controller by not a governance (50ms)
- ✓ shouldn't controller equal a zero address (55ms)

Contract: Governable Contract

✓ should setStorage correctly (269ms)

Contract: TokenGeyser Contract

TokenGeyser creating

✓ should create correctly (7199ms)

✓ should reward token be correct (39ms)

TokenGeyser staking

✓ should stake correctly (3833ms)

✓ shouldn't stake or unstake by not vault or controller (194ms)

✓ should unstake correctly (5714ms)

✓ should return rewardBalance correctly (11374ms)

✓ should unlock schedule count be correct (2842ms)

✓ should lock tokens correctly (2345ms)

✓ should unlock tokens correctly (2347ms)

✓ should rescue funds from unlocked pool correctly (2413ms)

✓ should rescue funds from locked pool correctly (2529ms)

Contract: TokenPool Contract

TokenPool details

✓ should balance of token pool be correctly (494ms)

✓ should transfer from token pool correctly (2018ms)

TokenPool rescueFunds

✓ should rescue funds correctly (666ms)

✓ shouldn't rescue held token by the contract (267ms)

✓ shouldn't rescue token by not an owner (863ms)

✓ shouldn't rescue token more that allow (757ms)

Contract: Vault Contract

Vault Contract creating phase

✓ should intialize correctly (1457ms)

✓ should controller be correct (755ms)

✓ should governance be correct (859ms)

✓ should set rewards distributor correctly (581ms)

✓ should set deposit fee correctly (639ms)

✓ should set vault fraction to invest correctly (4778ms)

✓ should set withdraw before reinvesting correctly (3926ms)

✓ should set allow share price decrease correctly (639ms)

```
    ✓ should  initialized vault only one time (235ms)
    ✓ shouldn't denominator be equal 0 (1187ms)
    ✓ shouldn't  invest more than 100% (2722ms)
    ✓ shouldn't  set rewards distributor by not a controller or governance (385ms)
    ✓ shouldn't  set vault fraction to invest by not a governance (4442ms)
    ✓ shouldn't denominator in vault fraction for investing equal 0 (3094ms)
    ✓ shouldn't numerator in vault fraction for investing be greater than denominator (296ms)
    ✓ shouldn't set deposit fee by not controller or governance (3733ms)
    ✓ should underlying balance in vault  be correct (3609ms)
  Vault Contract strategy
    ✓ should set strategy correctly (3070ms)
    ✓ shouldn't set already existed strategy (3269ms)
    ✓ shouldn't set strategy with zero address (2998ms)
    ✓ shouldn't set strategy with incorrect underlying (1932ms)
    ✓ shouldn't set strategy with incorrect vault (5191ms)
    ✓ should announce strategy update correctly (3894ms)
    ✓ should update strategy successfully (5131ms)
  Vault Contract start saving phase
    ✓ shouldn't  start saving when strategy is not defined (2575ms)
    ✓ should  start saving correctl (7260ms)
  Vault Contract deposit and withdraw phase
    ✓ should deposit correctly (7459ms)
    ✓ shouldn't rewardsDistributor  be empty while withdraw (5890ms)
    ✓ should depositFor and withdraw correctly (7786ms)
    ✓ shouldn't  withdraw when vault is empty (3416ms)
  Vault Contract upgrade vault phase
    ✓ should  schedule upgrade of vault correctly (4925ms)
    ✓ should upgrade vault correctly (7023ms)
    ✓ should finalize upgrade correctly (10138ms)

  124 passing (25m)
```

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts/	97.35	82.50	98.46	97.16	
Controllable.sol	100.00	50.00	100.00	100.00	
ControllableInit.sol	75.00	50.00	80.00	66.67	16, 20
Controller.sol	94.44	82.14	94.44	94.87	66, 109
Governable.sol	100.00	100.00	100.00	100.00	
GovernableInit.sol	100.00	75.00	100.00	100.00	
RewardToken.sol	100.00	100.00	100.00	100.00	
Storage.sol	100.00	100.00	100.00	100.00	
Vault.sol	97.08	85.94	100.00	97.12	179, 215, 464, 465
VaultProxy.sol	100.00	50.00	100.00	100.00	
VaultStorage.sol	100.00	100.00	100.00	100.00	
contracts/Interfaces/	100.00	100.00	100.00	100.00	
IController.sol	100.00	100.00	100.00	100.00	
IMigrator.sol	100.00	100.00	100.00	100.00	
IStrategy.sol	100.00	100.00	100.00	100.00	
IUpgradeSource.sol	100.00	100.00	100.00	100.00	
IVault.sol	100.00	100.00	100.00	100.00	
contracts/TokenDistribution/	95.43	70.59	100.00	95.00	
ClaimRewardProxy.sol	100.00	83.33	100.00	100.00	
IStaking.sol	100.00	100.00	100.00	100.00	
TokenGeyser.sol	94.89	68.33	100.00	100.00	... 725, 759, 774
TokenPool.sol	100.00	100.00	100.00	100.00	
contracts/compound/	100.00	100.00	100.00	100.00	
CTokenInterfaces.sol	100.00	100.00	100.00	100.00	

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
ComptrollerInterface.sol	100.00	100.00	100.00	100.00	
InterestRateModel.sol	100.00	100.00	100.00	100.00	
contracts/strategies/	100.00	89.71	100.00	100.00	
FundDivisionStrategy.sol	100.00	89.71	100.00	100.00	
contracts/strategies/aave/	94.74	88.89	100.00	97.22	
AToken.sol	100.00	100.00	100.00	100.00	
AaveStrategy.sol	94.74	88.89	100.00	97.22	... 725, 731, 734
LendingPool.sol	100.00	100.00	100.00	100.00	
contracts/strategies/compound/	89.58	73.68	96.55	90.63	
CompleteCToken.sol	100.00	100.00	100.00	100.00	
CompoundInteractor.sol	85.71	50.00	85.71	85.71	43, 54, 55
CompoundStrategy.sol	90.67	80.00	100.00	92.00	... 207, 211, 214
All files	**95.99**	**80.77**	**89.57**	**96.13**	

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

ZOKYO.