Security Assessment

NAOS Foundation I

Nov 25th, 2021

Table of Contents

Summary

This report has been prepared for NAOS Foundation to discover issues and vulnerabilities in the source code of the NAOS Foundation I project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Addtionally, this audit is based on a premise that all external contracts and calculation formula were implemented safely.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	NAOS Foundation I
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/NAOS-Finance/NAOS-Protocol/tree/master/contracts https://github.com/NAOS-Finance/NAOS-Galaxy/tree/develop/src
Commit	b41f8fbd1746567ba2c8144d70ce40c6bcb77cc8 264a024fdcc19d125de3a93b8f767d79d928b03a 358e38a6a99c2d1842c988f3dd999ada1e9d8e53 cec4fe843e34020e885f85d7331c69a97989ed01

Audit Summary

Delivery Date	Nov 25, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	1	0	0	1	0	0
Medium	0	0	0	0	0	0
Minor	2	0	0	1	0	1
Informational	24	0	0	0	0	24
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
NAS	contracts/NAOS-Galaxy/src/borrower/collect/collector.sol	6fcd95981584a87bbfe29ff063b84c5c522616f569664dabe40c708e47d6a4db
NAC	contracts/NAOS-Galaxy/src/borrower/fabs/collector.sol	4ca11af126178a4f07d0f561f9d742874cf17d7aaf588b9967995861ebb37cbd
NAK	contracts/NAOS-Galaxy/src/borrower/fabs/interfaces.sol	29fd51f04aa3eabed75b71c3e603b606241299e2c63e3ee028e969bf46757a99
NAP	contracts/NAOS-Galaxy/src/borrower/fabs/navfeed.sol	892241e5bb02ee6dd0de95b4c9017687639c114119db0cda6c5f0c3a5c7a281e
NOS	contracts/NAOS-Galaxy/src/borrower/fabs/nftfeed.sol	8703273c03797dc2667d963a44a2fc89b27faa626ea3b67c6014047857cf2908
NOG	contracts/NAOS-Galaxy/src/borrower/fabs/pile.sol	8032fbb6ab3ead70851b4947193d8c74a5540e6c413c654f9b26981514a77e27
NOC	contracts/NAOS-Galaxy/src/borrower/fabs/shelf.sol	289b88901b200a3cb2c92f0317b5718c789c58ad05536d2ac60b5072d525faf6
NOK	contracts/NAOS-Galaxy/src/borrower/fabs/title.sol	b1c3c0f917bcb5ef92a6f18a8b29b0462aefd2570d9a66ed960224fb8d8fbbbc
NOP	contracts/NAOS-Galaxy/src/borrower/feed/buckets.sol	225adbdb3c00d88f9576ca53981988a33b9e625584951f5252d2cd00b463d39d
NSG	contracts/NAOS-Galaxy/src/borrower/feed/navfeed.sol	3afe9e8f185dce443b930cb1ed26821f50e5f9823eb76ce84fc35ed2b4490d94
NSC	contracts/NAOS-Galaxy/src/borrower/feed/nftfeed.sol	aed13240a0c482a9923a877b4ab0a2903f7491d1aa2cf4119a8650439e7159a5
NSK	contracts/NAOS-Galaxy/src/borrower/deployer.sol	9620aca2bc1419f0c50a55c65ce3a99c23e0b4302bda14f9d4edb9db331d70cb
NSP	contracts/NAOS-Galaxy/src/borrower/pile.sol	62fb9cb38346cbf77e32cc29f93faf357e6474ae790aeaac217b96773d92092c
NGC	contracts/NAOS-Galaxy/src/borrower/shelf.sol	8d1269d6177f03f900cc69e13b02b1ef7356573cbd002f4a7d9cf2cd30ad3210

ID	File	SHA256 Checksum
NGP	contracts/NAOS-Galaxy/src/lender/fabs/assessor.sol	4424548ab99edf686402d3f82ca9e603c4608a9a1ef847b049b60972019a01c7
NCK	contracts/NAOS-Galaxy/src/lender/fabs/coordinator.sol	1d120dcc6746aa6368071b521e23bfcbf1d05a8dddc113cb7e416afb437d6e21
NCP	contracts/NAOS-Galaxy/src/lender/fabs/interfaces.sol	98f07573c84407081eadc3eac70c7a44b9964335271f63e84f9e11338d20e674
NKP	contracts/NAOS-Galaxy/src/lender/fabs/memberlist.sol	34d7ba7ae7cc95b864a4aa399ddecbe1f29e1685a280fc53aa5d1417862dbf2d
AOS	contracts/NAOS-Galaxy/src/lender/fabs/operator.sol	f86a7d59a282aa54062911ac031d921b05bea4d2071ecbdf6ee6a159f9e53333
AOG	contracts/NAOS-Galaxy/src/lender/fabs/reserve.sol	2a1888efd0a8ebe0cabe7bf31a93bd85afedec279aab892a672a44e05b2b3281
AOC	contracts/NAOS-Galaxy/src/lender/fabs/restrictedtoken.sol	63c866d4b9c95360b1ccc32a06db6c9724b921289e08c75369377b323d2eeae8
AOK	contracts/NAOS-Galaxy/src/lender/fabs/tranche.sol	436c5f17217476d013d59e345f21ec8ee501e55b90c45c95162ef3d344b2d388
ASG	contracts/NAOS-Galaxy/src/lender/token/memberlist.sol	068c1dccb74e79872b872a2c4fb4bab6441b4f8dfe5b7ced3141aec2464523f3
ASC	contracts/NAOS-Galaxy/src/lender/token/restricted.sol	187057891212cf83437d5fe2851697e410001769bf9c828953b5cbd1b4b50574
ASK	contracts/NAOS-Galaxy/src/lender/assessor.sol	a2a9f62f44e00cfc4a00f9731e638167fe1d362dfe311d9a6eb9e97fbd1b6fe9
ASP	contracts/NAOS-Galaxy/src/lender/coordinator.sol	28b3432bbe4bf8d5a3258878d4491355b7f28aa556a91076d67a2c3f9dae4476
AGC	contracts/NAOS-Galaxy/src/lender/deployer.sol	b54db48c5671956d31c1ccc55adcfa20eb93d04b2fbdd52bc4be19a21d813e1d
AGK	contracts/NAOS-Galaxy/src/lender/operator.sol	34952367853606c4bc0bd12fed0905749cdb70b0d5f1b37d309bcacfd48bd919
AGP	contracts/NAOS-Galaxy/src/lender/reserve.sol	eb7de19a446736a53a717ce0f10b0ff4ef26819be94fbd5d80f04e75a27dbbf8

ID	File	SHA256 Checksum
ACK	contracts/NAOS-Galaxy/src/lender/tranche.sol	4ffee4f7811970f1ebe9cc7dce7bd4f2e0e4bf6e28cda2a7d72a28b6cf48d392
ACP	contracts/NAOS-Galaxy/src/fixed_point.sol	6faa197d612f99e34ddc36fba9184cd77a7b856be193d22c7fdcd1e0e399a120
AKP	contracts/NAOS-Galaxy/src/root.sol	14aac76aa76b153fe4ed40300fa2bafd5201460d31e4e51900f954bdacfe6bed
YVA	contracts/NAOS-Protocol/contracts/Staking/adapters/YearnVaultAdapter.sol	4e5f774f683639745099ee0c19618b0b2e972827d36e8fd01eb9ac03a49fa7c9
VIS	contracts/NAOS-Protocol/contracts/Staking/libraries/betaInsurance/Vault.sol	9ae50b907e82f94ca0a49580f295c478615558322e82607042eba1c9f69b87e4
PPS	contracts/NAOS-Protocol/contracts/Staking/libraries/boostPools/Pool.sol	73184e85d611895bf3854197ce4982c8a5033221b75e779803ecc1d4810c2808
SPS	contracts/NAOS-Protocol/contracts/Staking/libraries/boostPools/Stake.sol	a496d589ba88ba87f249f3a5354e74495a325c4af8eaa88b265c3070aa898d87
PSA	contracts/NAOS-Protocol/contracts/Staking/libraries/pools/Pool.sol	17340a3dd44ac22ef9ea3fd4c9ee991471fe723ae44e20671cb8a84523924841
SSN	contracts/NAOS-Protocol/contracts/Staking/libraries/pools/Stake.sol	a496d589ba88ba87f249f3a5354e74495a325c4af8eaa88b265c3070aa898d87
PWT	contracts/NAOS-Protocol/contracts/Staking/libraries/poolsWithTransfer/Pool.sol	b6b63a3ce06e01579d959555b11641d9595e928af1b53fbf4cb50baef92dbf96
SWT	contracts/NAOS-Protocol/contracts/Staking/libraries/poolsWithTransfer/Stake.sol	918da5fcb8fd652ae5cdbf7b267ed9e2323880ebae793c992916a856e55e18dc
FPM	contracts/NAOS-Protocol/contracts/Staking/libraries/FixedPointMath.sol	f031f1f809c6676129dc03aa89042642c359bee9bbe851801ab20585c6634d00
BIS	contracts/NAOS-Protocol/contracts/Staking/BetaInsurance.sol	81037684869059411489ba1a9d850e2cedcbe9af23d22fef549b535256b21a93
BPS	contracts/NAOS-Protocol/contracts/Staking/BoostPool.sol	28ebe0db399482c249b84144e561d8f78a9c944883f99055d0b891974420fa2d
GSP	contracts/NAOS-Protocol/contracts/Staking/GalaxyStakingPools.sol	821cfd14a20c28c92e04f676cd0a5e17fb43bea0a91917cecab503924ee6d86f

ID	File	SHA256 Checksum
SPW	contracts/NAOS-Protocol/contracts/Staking/StakingPoolsWithTransfer.sol	974319c4da6019019bf7336325f19f7f1ce2475bad04323d4ede00ecc7349bd6

Privileged Functions

The contract contains the following privileged functions that are restricted by some modifiers. They are used to modify the contract configurations and address attributes. We grouped these functions below:

NAOS-PROTOCOL

The `expectVaultInitialized` modifier:

Contract `BetaInsurance`:

- flushActiveVault()
- harvest(uint256 _vaultId)
- recallFundsFromVault(uint256 _vaultId, uint256 _amount)

The `beforePaymentCheck` modifier:

Contract `BetaInsurance`:

- payPremiumByCurrency(uint256 _insuranceID, uint256 _naosAmountOutMin)
- payPremiumByNAOS(uint256 _insuranceID)

The `onlyAdmin` modifier:

Contract `YearnVaultAdapter`:

- deposit(uint256 _amount)
- withdraw(address _recipient, uint256 _amount)

The `onlyAdmins` modifier:

Contract `BetaInsurance`:

- harvest(uint256 _vaultId)

Contract `GalaxyStakingPools`:

- setWhitelist(address _user, bool _state)

The `onlyGov` modifier:

Contract `GalaxyStakingPools`:

- setPendingGovernance(address _pendingGovernance)
- setAdmin(address _user, bool _state)
- setRewardRate(uint256 _rewardRate)
- setRewardWeights(uint256[] calldata _rewardWeights)
- setDepositedCeiling(uint256 _poolId, uint256 _amount)
- createPool(uint256 _expiredTimestamp)

The `onlyWhitelist` modifier:

Contract `GalaxyStakingPools`:

- deposit(uint256 _poolId, uint256 _amount)

The `onlyUpdated` modifier:

Contract `GalaxyStakingPools`:

- deposit(uint256 _poolId, uint256 _amount)
- redeem(uint256 _poolId, uint256[] calldata _index)
- withdraw()
- claim(uint256 _poolId)
- activateBoost(uint256 _poolId, address _account)
- activateBoosts(address _account)

The `nonReentrant` modifier:

Contract `BoostPool`:

- deposit(uint256 _depositAmount, uint256 _index)
- withdraw(uint256[] calldata _index)
- claimImmediately()
- claim()
- startCoolDown()
- donateReward(uint256 _donateAmount)

Contract `GalaxyStakingPools`:

- deposit(uint256 _poolId, uint256 _amount)
- redeem(uint256 _poolId, uint256[] calldata _index)
- withdraw()
- claim(uint256 _poolId)

- activateBoost(uint256 _poolId, address _account)
- activateBoosts(address _account)

Contract `StakingPoolsWithTransfer`:

- deposit(uint256 _poolId, uint256 _depositAmount)
- withdraw(uint256 _poolId, uint256 _withdrawAmount)
- claim(uint256 _poolId)
- exit(uint256 _poolId)
- donateReward(uint256 _poolId, uint256 _donateAmount)

The `onlyGovernance` modifier:

Contract `BetaInsurance`:

- setPendingGovernance(address _pendingGovernance)
- setAdmin(address _user, bool _state)
- setStakingPool(IStakingPoolWithTransfer _stakingPool, uint256 _poolId)
- setTransmuter(IBetaTransmuter _transmuter)
- updateActiveVault(IVaultAdapter _adapter)
- setHarvestFee(uint256 _harvestFee)
- setInsurancePremium(uint256 _insuranceID, uint256 _premiumCurrencyAmount, uint256 _premiumNAOSAmount)
- compensate(uint256 _insuranceID, uint256 _amount)

Contract `BoostPool`:

- setPendingGovernance(address _pendingGovernance)
- setRewardRate(uint256 _rewardRate)
- setLockTimeWeighted(uint256 _lockTime, uint256 _weighted)
- setCooldown(uint256 _cooldownPeriod)
- setPenaltyPercent(uint256 _penaltyPercent)

Contract `StakingPoolsWithTransfer`:

- setPendingGovernance(address _pendingGovernance)
- setRewardRate(uint256 _rewardRate)
- createPool(IERC20 _token)
- setRewardWeights(uint256[] calldata _rewardWeights)

NAOS-GALAXY

The `auth_collector` modifier:

Contract `Collector`:

- collect(uint256 loan)

The `auth` modifier:

Contract `Collector`:

- relyCollector(address usr)
- denyCollector(address usr)
- depend(bytes32 contractName, address addr)
- file(bytes32 what, uint256 loan, address buyer, uint256 nftPrice)
- collect(uint256 loan, address buyer)

Contract `NAVFeed`:

- file(bytes32 name, uint256 risk_, uint256 thresholdRatio_, uint256 ceilingRatio_, uint256 rate_, uint256 recoveryRatePD_)
- file(bytes32 name, bytes32 nftID_, uint256 maturityDate_)
- file(bytes32 name, uint256 value)
- borrow(uint256 loan, uint256 amount)
- update(bytes32 nftID_, uint256 value, uint256 risk_)
- repay(uint256 loan, uint256 amount)

Contract `NAVFeed`:

- depend(bytes32 contractName, address addr)
- file(bytes32 name, uint256 risk_, uint256 thresholdRatio_, uint256 ceilingRatio_, uint256 rate_)
- update(bytes32 nftID_, uint256 value)
- update(bytes32 nftID_, uint256 value, uint256 risk_)
- borrow(uint256 loan, uint256 amount)
- repay(uint256, uint256 amount)
- borrowEvent(uint256 loan)

Contract `Pile`:

- incDebt(uint256 loan, uint256 currencyAmount)

- decDebt(uint256 loan, uint256 currencyAmount)
- setRate(uint256 loan, uint256 rate)
- changeRate(uint256 loan, uint256 newRate)
- file(bytes32 what, uint256 rate, uint256 value)

Contract `Shelf`:

- depend(bytes32 contractName, address addr)
- recover(
  uint256 loan, address usr, uint256 currencyAmount)
- claim(uint256 loan, address usr)

Contract `Memberlist`:

- updateMember(address usr, uint256 validUntil)
- updateMembers(address[] memory users, uint256 validUntil)

Contract `RestrictedToken`:

- depend(bytes32 contractName, address addr)

Contract `Assessor`:

- depend(bytes32 contractName, address addr)
- file(bytes32 name, uint256 value)
- changeSeniorAsset(uint256 seniorRatio_, uint256 seniorSupply, uint256 seniorRedeem)
- repaymentUpdate(uint256 currencyAmount)
- borrowUpdate(uint256 currencyAmount)

Contract `EpochCoordinator`:

- file(bytes32 name, uint256 value)
- depend(bytes32 contractName, address addr)
- closeEpoch()
- submitSolution(uint256 seniorRedeem, uint256 juniorRedeem, uint256 juniorSupply, uint256 seniorSupply)
- executeEpoch()

Contract `Operator`:

- depend(bytes32 contractName, address addr)

Contract `Reserve`:

- file(bytes32 what, uint256 amount)
- depend(bytes32 contractName, address addr)
- deposit(uint256 currencyAmount)
- payout(uint256 currencyAmount)
- payoutTo(address to, uint256 currencyAmount)

Contract `Tranche`:

- depend(bytes32 contractName, address addr)
- supplyOrder(address usr, uint256 newSupplyAmount)
- redeemOrder(address usr, uint256 newRedeemAmount)
- disburse(address usr)
- disburse(address usr, uint256 endEpoch)
- epochUpdate(uint256 epochID, uint256 supplyFulfillment_, uint256 redeemFulfillment_, uint256 tokenPrice_, uint256 epochSupplyOrderCurrency, uint256 epochRedeemOrderCurrency)
- closeEpoch()
- mint(address usr, uint256 amount)
- authTransfer(address erc20, address usr, uint256 amount)

Contract `GalaxyRoot`:

- relyContract(address target, address usr)
- denyContract(address target, address usr)
- file(bytes32 name, address payable usr)
- withdrawFee(uint currencyAmount)

The `orderAllowed(address usr)` modifier:

Contract `Tranche`:

- supplyOrder(address usr, uint256 newSupplyAmount)
- redeemOrder(address usr, uint256 newRedeemAmount)

The `minimumEpochTimePassed` modifier:

Contract `Collector`:

- closeEpoch()

The `owner(loan)` modifier:

Contract `Shelf`:

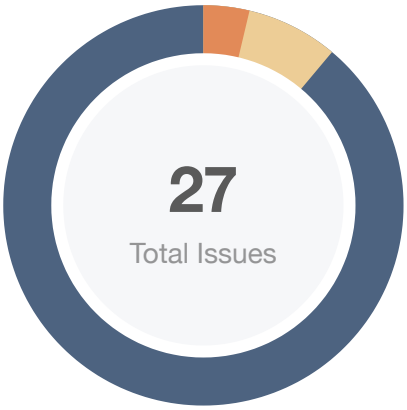
- borrow(uint256 loan, uint256 currencyAmount)
- withdraw(uint256 loan, uint256 currencyAmount, address usr)
- repay(uint256 loan, uint256 currencyAmount)
- lock(uint256 loan)
- unlock(uint256 loan)

The `owner(loan)` modifier:

Contract `RestrictedToken`:

- depend(bytes32 contractName, address addr)

Findings



Critical	0 (0.00%)
Major	1 (3.70%)
Medium	0 (0.00%)
Minor	2 (7.41%)
Informational	24 (88.89%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Unlocked Compiler Version Declaration	Language Specific	Informational	Resolved
GLOBAL-02	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
ACK-01	Function Visibility Optimization	Gas Optimization	Informational	Resolved
ACK-02	Boolean Equality	Coding Style	Informational	Resolved
AGK-01	Function Visibility Optimization	Gas Optimization	Informational	Resolved
AGK-02	Boolean Equality	Coding Style	Informational	Resolved
AGK-03	Meaningless Code	Coding Style	Informational	Resolved
AGP-01	Function Visibility Optimization	Gas Optimization	Informational	Resolved
AGP-02	Check Effect Interaction Pattern Violated	Logical Issue	Informational	Resolved
AKP-01	Function Visibility Optimization	Gas Optimization	Informational	Resolved
AKP-02	Missing Input Validation	Logical Issue	Informational	Resolved
AKP-03	Boolean Equality	Coding Style	Informational	Resolved
ASK-01	Function Visibility Optimization	Gas Optimization	Informational	Resolved

ID	Title	Category	Severity	Status
ASK-02	Return Value Optimization	Logical Issue, Coding Style	Informational	Resolved
ASP-01	Boolean Equality	Coding Style	Informational	Resolved
ASP-02	Missing Input Validation	Logical Issue	Minor	Acknowledged
BIS-01	Check Effect Interaction Pattern Violated	Logical Issue	Informational	Resolved
BIS-02	Function Visibility Optimization	Gas Optimization	Informational	Resolved
BPS-01	Function Visibility Optimization	Gas Optimization	Informational	Resolved
GSP-01	Function Visibility Optimization	Gas Optimization	Informational	Resolved
NAS-01	Function Visibility Optimization	Gas Optimization	Informational	Resolved
NGC-01	Function Visibility Optimization	Gas Optimization	Informational	Resolved
NSC-01	Missing Input Validation	Logical Issue	Minor	Resolved
NSC-02	Function Visibility Optimization	Gas Optimization	Informational	Resolved
NSG-01	Function Visibility Optimization	Gas Optimization	Informational	Resolved
NSG-02	Remove Test Case	Coding Style	Informational	Resolved
PPS-01	Meaningless State Variables	Coding Style	Informational	Resolved

GLOBAL-01 | Unlocked Compiler Version Declaration

Category	Severity	Location	Status
Language Specific	Informational	Global	Resolved

Description

The compiler version utilized throughout the project uses the `>=` and `<` prefix specifier, denoting that a compiler version that is greater than the version will be used to compile the contracts. It is recommend the compiler version should be consistent throughout the codebase.

Recommendation

It is a general practice to alternatively lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and in so doing be able to identify emerging ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities required by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01 and 358e38a6a99c2d1842c988f3dd999ada1e9d8e53.

GLOBAL-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	Global	Acknowledged

Description

Naos-Protocol:
the `governance` has the authority over the following function:
Contract `BetaInsurance`:

- setPendingGovernance(address _pendingGovernance)
- setAdmin(address _user, bool _state)
- setStakingPool(IStakingPoolWithTransfer _stakingPool, uint256 _poolId)
- setTransmuter(IBetaTransmuter _transmuter)
- updateActiveVault(IVaultAdapter _adapter)
- setHarvestFee(uint256 _harvestFee)
- setInsurancePremium(uint256 _insuranceID, uint256 _premiumCurrencyAmount, uint256 _premiumNAOSAmount)
- compensate(uint256 _insuranceID, uint256 _amount)

Contract `BoostPool`:

- setPendingGovernance(address _pendingGovernance)
- setRewardRate(uint256 _rewardRate)
- setLockTimeWeighted(uint256 _lockTime, uint256 _weighted)
- setCooldown(uint256 _cooldownPeriod)
- setPenaltyPercent(uint256 _penaltyPercent)

Contract `StakingPoolsWithTransfer`:

- setPendingGovernance(address _pendingGovernance)
- setRewardRate(uint256 _rewardRate)
- createPool(IERC20 _token)
- setRewardWeights(uint256[] calldata _rewardWeights)

Contract `GalaxyStakingPools`:

- setPendingGovernance(address _pendingGovernance)
- setAdmin(address _user, bool _state)

- setRewardRate(uint256 _rewardRate)
- setRewardWeights(uint256[] calldata _rewardWeights)
- setDepositedCeiling(uint256 _poolId, uint256 _amount)
- createPool(uint256 _expiredTimestamp)

the `admin` has the authority over the following function:
Contract `GalaxyStakingPools`:

- setWhitelist(address _user, bool _state)

Naos-Galaxy:
the `auth` account has the authority over the following function:
Contract `Tranche`:

- authTransfer(address erc20, address usr, uint256 amount)
- mint(address usr, uint256 amount)

Contract `GalaxyRoot`:

- withdrawFee(uint currencyAmount) without obtaining the consensus of the community.

Recommendation

We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked.

In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The client response:

After deployment, the governance role will be transferred to the NAOS multisig account which consists of NAOS core team members and trusted community members. The setting will follow the decision of the NAOS governance process.

ACK-01 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/NAOS-Galaxy/src/lender/tranche.sol: 115, 128, 149, 235, 298, 331, 413	Resolved

Description

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays, `external` functions are more efficient than `public` functions.

Recommendation

We advise the client to use the external attribute for functions never called from the contract.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01.

ACK-02 | Boolean Equality

Category	Severity	Location	Status
Coding Style	Informational	contracts/NAOS-Galaxy/src/lender/tranche.sol: 306, 332	Resolved

Description

Detects the comparison to boolean constants. Boolean constants can be used directly and do not need to be compare to true or false.

Recommendation

We recommend removing the equality to the boolean constant. For example:

```
bool A = true;
bool B = false;
require(A, "A must be true!");
require(!B, "B must be false!")
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01.

AGK-01 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/NAOS-Galaxy/src/lender/operator.sol: 86, 142, 154, 166	Resolved

Description

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays, `external` functions are more efficient than `public` functions.

Recommendation

We advise the client to use the external attribute for functions never called from the contract.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01.

AGK-02 | Boolean Equality

Category	Severity	Location	Status
Coding Style	Informational	contracts/NAOS-Galaxy/src/lender/operator.sol: 96, 102, 117, 130	Resolved

Description

Detects the comparison to boolean constants. Boolean constants can be used directly and do not need to be compare to true or false.

Recommendation

We recommend removing the equality to the boolean constant. For example:

```
bool A = true;
bool B = false;
require(A, "A must be true!");
require(!B, "B must be false!")
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01.

AGK-03 | Meaningless Code

Category	Severity	Location	Status
Coding Style	Informational	contracts/NAOS-Galaxy/src/lender/operator.sol: 103	Resolved

Description

`token.hasMember(msg.sender)` is called twice and the return value of the second call is not handled.

Recommendation

We recommend removing the meaningless code.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01.

AGP-01 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/NAOS-Galaxy/src/lender/reserve.sol: 56, 62, 77, 87, 92, 103	Resolved

Description

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays, `external` functions are more efficient than `public` functions.

Recommendation

We advise the client to use the external attribute for functions never called from the contract.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01.

AGP-02 | Check Effect Interaction Pattern Violated

Category	Severity	Location	Status
Logical Issue	Informational	contracts/NAOS-Galaxy/src/lender/reserve.sol: 82, 97	Resolved

Description

The order of external call/transfer and storage manipulation must follow the check-effect-interaction pattern.

Recommendation

We advise the client to check if storage manipulation is before the external call/transfer operation.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01.

AKP-01 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/NAOS-Galaxy/src/root.sol: 65, 80, 113, 122	Resolved

Description

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays, `external` functions are more efficient than `public` functions.

Recommendation

We advise the client to use the external attribute for functions never called from the contract.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01.

AKP-02 | Missing Input Validation

Category	Severity	Location	Status
Logical Issue	Informational	contracts/NAOS-Galaxy/src/root.sol: 60	Resolved

Description

The given input is missing the sanity check for non-zero address in the aforementioned line.

Recommendation

We recommend adding the check for the passed-in values to prevent unexpected error as below: constructor():

```
60  require(deployUsr_ != address(0), "deployUsr_ address cannot be 0");
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01.

AKP-03 | Boolean Equality

Category	Severity	Location	Status
Coding Style	Informational	contracts/NAOS-Galaxy/src/root.sol: 81	Resolved

Description

Detects the comparison to boolean constants. Boolean constants can be used directly and do not need to be compare to true or false.

Recommendation

We recommend removing the equality to the boolean constant. For example:

```
bool A = true;
bool B = false;
require(A, "A must be true!");
require(!B, "B must be false!")
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01.

ASK-01 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/NAOS-Galaxy/src/lender/assessor.sol: 77, 89, 189, 208	Resolved

Description

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays, `external` functions are more efficient than `public` functions.

Recommendation

We advise the client to use the external attribute for functions never called from the contract.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01.

ASK-02 | Return Value Optimization

Category	Severity	Location	Status
Logical Issue, Coding Style	Informational	contracts/NAOS-Galaxy/src/lender/assessor.sol: 244	Resolved

Description

`now` is always greater than `lastUpdateSeniorInterest`, so just return the calculation result of `chargeInterest()` or remove the `=` in the judgment condition.

Recommendation

We recommend modifying as below:

```
244    function seniorDebt() public view returns (uint256) {
245        if (now > lastUpdateSeniorInterest) {
246            return chargeInterest(seniorDebt_, seniorInterestRate.value,
lastUpdateSeniorInterest);
247        }
248        return seniorDebt_;
249    }
```

or

```
244    function seniorDebt() public view returns (uint256) {
245        return chargeInterest(seniorDebt_, seniorInterestRate.value,
lastUpdateSeniorInterest);
246    }
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01.

ASP-01 | Boolean Equality

Category	Severity	Location	Status
Coding Style	Informational	contracts/NAOS-Galaxy/src/lender/coordinator.sol: 198, 275, 306, 327, 417, 552	Resolved

Description

Detects the comparison to boolean constants. Boolean constants can be used directly and do not need to be compare to true or false.

Recommendation

We recommend removing the equality to the boolean constant. For example:

```
bool A = true;
bool B = false;
require(A, "A must be true!");
require(!B, "B must be false!")
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01.

ASP-02 | Missing Input Validation

Category	Severity	Location	Status
Logical Issue	Minor	contracts/NAOS-Galaxy/src/lender/coordinator.sol: 269	Acknowledged

Description

According to logic, after the first proposal that meets the requirements is submitted, it enters the challenge period. During the challenge period, the proposal can be submitted again, so the submission time should be verified.

Recommendation

We recommend modifying as below:

```
269 function submitSolution(
270         uint256 seniorRedeem,
271         uint256 juniorRedeem,
272         uint256 juniorSupply,
273         uint256 seniorSupply
274     ) public auth returns (int256) {
275         require(block.timestamp < minChallengePeriodEnd, "challenge-period-end");
276         ...
277 }
```

Alleviation

The client response:
If we added this validation, the system might stuck when we didn't submit the solution after the challenge time. The function could only be undertaken by the admin. After the evaluation for trade-off, we will not make the change according to it.

BIS-01 | Check Effect Interaction Pattern Violated

Category	Severity	Location	Status
Logical Issue	Informational	contracts/NAOS-Protocol/contracts/Staking/BetaInsurance.sol: 261, 543	Resolved

Description

The order of external call/transfer and storage manipulation must follow the check-effect-interaction pattern.

Recommendation

We advise the client to check if storage manipulation is before the external call/transfer operation.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 358e38a6a99c2d1842c988f3dd999ada1e9d8e53.

BIS-02 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/NAOS-Protocol/contracts/Staking/BetaInsurance.sol: 178, 259, 277	Resolved

Description

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays, `external` functions are more efficient than `public` functions.

Recommendation

We advise the client to use the external attribute for functions never called from the contract.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 358e38a6a99c2d1842c988f3dd999ada1e9d8e53.

BPS-01 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/NAOS-Protocol/contracts/Staking/BoostPool.sol: 276	Resolved

Description

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays, `external` functions are more efficient than `public` functions.

Recommendation

We advise the client to use the external attribute for functions never called from the contract.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 358e38a6a99c2d1842c988f3dd999ada1e9d8e53.

GSP-01 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/NAOS-Protocol/contracts/Staking/GalaxyStakingPools.sol: 181	Resolved

Description

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays, `external` functions are more efficient than `public` functions.

Recommendation

We advise the client to use the external attribute for functions never called from the contract.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 358e38a6a99c2d1842c988f3dd999ada1e9d8e53.

NAS-01 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/NAOS-Galaxy/src/borrower/collect/collector.sol: 60, 64	Resolved

Description

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays, `external` functions are more efficient than `public` functions.

Recommendation

We advise the client to use the external attribute for functions never called from the contract.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01.

NGC-01 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/NAOS-Galaxy/src/borrower/shelf.sol: 283	Resolved

Description

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays, `external` functions are more efficient than `public` functions.

Recommendation

We advise the client to use the external attribute for functions never called from the contract.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01.

NSC-01 | Missing Input Validation

Category	Severity	Location	Status
Logical Issue	Minor	contracts/NAOS-Galaxy/src/borrower/feed/nftfeed.sol: 116	Resolved

Description

`thresholdRatio` is the critical value that triggers asset liquidation, `ceilingRatio` represents the ratio of the user's loan amount to the staked assets, so `thresholdRatio` should be greater than `ceilingRatio`.

Recommendation

We advise adding the check for the passed-in values to prevent unexpected error as below:

```
124    require(thresholdRatio_ > ceilingRatio_, "thresholdRatio_ must be greater than ceilingRatio_");
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01.

NSC-02 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/NAOS-Galaxy/src/borrower/feed/nftfeed.sol: 93, 137, 143, 177, 189	Resolved

Description

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays, `external` functions are more efficient than `public` functions.

Recommendation

We advise the client to use the external attribute for functions never called from the contract.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01.

NSG-01 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/NAOS-Galaxy/src/borrower/feed/navfeed.sol: 66, 174, 382	Resolved

Description

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays, `external` functions are more efficient than `public` functions.

Recommendation

We advise the client to use the external attribute for functions never called from the contract.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01.

NSG-02 | Remove Test Case

Category	Severity	Location	Status
Coding Style	Informational	contracts/NAOS-Galaxy/src/borrower/feed/navfeed.sol: 66	Resolved

Description

After comments and communication, we learned that part of the initialization of `raskGroup` in the `init()` is the test case. There is no need to keep this part of the code after testing.

Recommendation

We recommend removing the test case to keep the code concise and avoid unexpected errors.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit cec4fe843e34020e885f85d7331c69a97989ed01.

PPS-01 | Meaningless State Variables

Category	Severity	Location	Status
Coding Style	Informational	contracts/NAOS-Protocol/contracts/Staking/libraries/boostPools/Pool.sol: 21	Resolved

Description

The variables are never used in the aforementioned line.

Recommendation

We recommend removing variables that have never been used.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 358e38a6a99c2d1842c988f3dd999ada1e9d8e53.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

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

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.