HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Colony Lab LTD
Date: October 24th, 2022

This report may contain confidential information about IT systems and the intellectual property of the Customer, as well as information about potential vulnerabilities and methods of their exploitation.

The report can be disclosed publicly after prior consent by another Party. Any subsequent publication of this report shall be without mandatory consent.

Document

Name	Smart Contract Code Review and Security Analysis Report for Colony Lab LTD
Approved By	Evgeniy Bezuglyi | SC Audits Department Head at Hacken OU
Type	Access; Staking; Vesting; Project Factory; ERC20
Platform	EVM
Network	Avalanche C-chain
Language	Solidity
Methods	Manual Review, Automated Review, Architecture Review
Website	www.colonylab.io
Timeline	12.09.2022 - 24.10.2022
Changelog	30.09.2022 - Initial Review 24.10.2022 - Second Review

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Colony Lab LTD (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

Scope

The scope of the project is smart contracts in the repository:

Initial review scope

Repository:
    https://github.com/ColonyLab/colony-app/tree/develop
Commit:
    4107b6b5ea5e5df354ca498138f9b8fc50ae9c35
Documentation: Yes

Integration and Unit Tests: Yes
Contracts:

```
    File: ./contracts/Access/KYCManager.sol
    SHA3: c546e40719fab1c4cb900101d1611e71b517ed62e38e2c1420902117339f13e8

    File: ./contracts/Access/MasterACL.sol
    SHA3: ffc5b5aff1749a6d03a195eb3d44101e5a906de17193785a6f9891cb1fb25c26

    File: ./contracts/Access/PrivilegedGroup.sol
    SHA3: fd57d817c115d3cd75befea12f9f3642ea20232a50a06fd7fc95b939bfadf75f

    File: ./contracts/Access/PrivilegedGroupUpgradeable.sol
    SHA3: 33726593b695c7babe7665956ab73e1c2e71860b0e77a9568865e5ed81c86ab2

    File: ./contracts/EarlyStage/AnalysisManager.sol
    SHA3: 2bae1019508cece07ca2c2b02604a7ad048fe6acdd92f013d293c33d07c44cc8

    File: ./contracts/EarlyStage/CommentManager.sol
    SHA3: 26b1f1402460f3ec5604398bc65499519b410f3a96412602d1d5932fccc25b75

    File: ./contracts/EarlyStage/CountdownManager.sol
    SHA3: bd68f9b1f0426215bcb2e6caa2c6b34de9df727475dff7dd6939d96ab69c92ad

    File: ./contracts/EarlyStage/EarlyStageManager.sol
    SHA3: a93c6904e81b9675157e534a4263d4be5cc48682498426a3fe4815d0694a9085

    File: ./contracts/EarlyStage/EventStoredList.sol
    SHA3: f8c7c9debcf7c014431c00731816cd5dbc84024cea9af7a527b38e43530af499

    File: ./contracts/EarlyStage/ProjectNest.sol
    SHA3: 1ff0b8d0ec562744496c4b026f7fbd084d2090d47bdf0b120279d447defc2f5a

    File: ./contracts/EarlyStage/ProjectNestFactory.sol
    SHA3: b00bb8c2d445724662ce066f566a0d3605549ba79c5b53093945091b3907af62

    File: ./contracts/EarlyStage/UpvoteManager.sol
    SHA3: 6426e7dec59cadeb35d038f8360cc53b1d46f3efb2517ed4afb476e839c63836
```

```
File: ./contracts/Interfaces/IAnalysisManager.sol
SHA3: 03a40e8a0b389f8ed0c926b120798f29dc9508d4bdcac97ededcc0af823589bc

File: ./contracts/Interfaces/ICeTokenDistributionStrategy.sol
SHA3: e1e7fadcc2d9143c352eee43760769496d1f642a47e838cbd456d550935bf0e5

File: ./contracts/Interfaces/ICeTokenFactory.sol
SHA3: 230139b6abb6f225825de770aeb08bb309f26cddc6b33d112bc7cd9c99f969d8

File: ./contracts/Interfaces/ICollateralToken.sol
SHA3: a588cb669cd14008682b2e0ffeace84a493e98ecac7c89c44061f96af1ea0901

File: ./contracts/Interfaces/IDiscreteVestingV2.sol
SHA3: c52d1925e08a95c175c5fbd2700d504e3b4be2be4389e2997b20e5868205e15f

File: ./contracts/Interfaces/IDiscreteVestingV2Factory.sol
SHA3: 3084e9004551ebea88ab0d7d63c1bfe003772ef1a870db8468f63e228541afb0

File: ./contracts/Interfaces/IEarlyStageManager.sol
SHA3: 51fbc9b7e89f340a022f934180bfad2d7b301ebd06b41ade0e28302a12884c05

File: ./contracts/Interfaces/IKYC.sol
SHA3: 0719da9b6f5ae73a250f894633e7c8be9b3357f8c2dc311e6e5f49ebd8d37847

File: ./contracts/Interfaces/ILinearVestingV2.sol
SHA3: d5f636652f4dac8979456ddb01bc405ace6cb4e653d4e3ebb12f3657730bfbcf

File: ./contracts/Interfaces/ILinearVestingV2Factory.sol
SHA3: 00b9bf8243a792a63bb311b5f25694e7c235fae564865dcc0709f77d641b9851

File: ./contracts/Interfaces/IMasterACL.sol
SHA3: efe7305ab9200745df31745814efd66704f6a56046668664b97f4cc80b14f30a

File: ./contracts/Interfaces/IProjectNest.sol
SHA3: 5f3df47efbc7b7f9c00b1d614eafbe395624fd9ab5cd6537d2fa0e93ee03897a

File: ./contracts/Interfaces/IProjectNestFactory.sol
SHA3: 6750fb4dc2588802bbd0bc48d5edad9c99639cbe49a0e003b858023109524af9

File: ./contracts/Interfaces/IStakingV2Auth.sol
SHA3: dde4f1adde7c25493ff3f1ac711bc6c4730b3dfbaa1ad0e064ae8b1a20a294d3

File: ./contracts/Interfaces/IStakingV2RewardNotifier.sol
SHA3: 44055ea263ff04f55a6418bcfe6c3bb7397e67b550f764bbcfd9d9bec3fe5796

File: ./contracts/Interfaces/IVestingV2.sol
SHA3: 59fe8a6f0bc962567375bbf5fa2d450fc8a37149db2f9dc23b0caaf47e2884ce

File: ./contracts/StakingV2/AntToken.sol
SHA3: 766fb45e4036d75115520d951fdddc009ae41fe7449cdeaf6e25b8839770182b

File: ./contracts/StakingV2/StakingV2.sol
SHA3: 664db5a10b2de0123c0dfb9de6a45cb6365079134a8801ffdfb861737d619da6

File: ./contracts/StakingV2/UniversalClaimer.sol
SHA3: 8fab7c97c1c62d11953704f18c9526ae48562efab3e7afb82b8a3e195336a78d

File: ./contracts/VestingV2/AbstractVestingV2.sol
SHA3: 14f6616ee08b475ab59f49e6cfdd143d896f1ad8389112d06d66e6f3df830698
```

```
File: ./contracts/VestingV2/AbstractVestingV2.sol
SHA3: 14f6616ee08b475ab59f49e6cfdd143d896f1ad8389112d06d66e6f3df830698

File: ./contracts/VestingV2/ceToken.sol
SHA3: 152f427c0801097600388b6d4e864269208ab2ba250a7cba4225a215a303b469

File: ./contracts/VestingV2/ceTokenDistributor.sol
SHA3: 8c19450e1d4cddff949722973776aed7c9bf5b5793759af4566a8112527de873

File: ./contracts/VestingV2/ceTokenFactory.sol
SHA3: 7b410251a930bb8e673ab45dead30f8e174bbafbca598f6f79aae06e3ee77c15

File: ./contracts/VestingV2/DiscreteVestingV2.sol
SHA3: 97761b8df81ec720286b8391434495a9cc984cf03ae60a5e662c1b3b5f3ae2a0

File: ./contracts/VestingV2/DiscreteVestingV2Factory.sol
SHA3: 09bfe33660538dc875f91aecdf16b346c83999e52934c36a409f5ff4e5ce0bbe

File: ./contracts/VestingV2/LinearVestingV2.sol
SHA3: 5badd6451839a2f0a08cf68401489aa902397c39906e1b0b35fa318aa94e18e5

File: ./contracts/VestingV2/LinearVestingV2Factory.sol
SHA3: 9ab4ab77fa26ca8559128e1e1cc12146833f2481df3b4137f6e62ddbf238d535
```

Second review scope

Repository:

https://github.com/ColonyLab/colony-app/tree/develop

Commit:

911229a5c6ca5ce95b6faf5b9b5e90d3ef8996f7

Documentation: Yes

Integration and Unit Tests: Yes

Contracts:

```
File: ./contracts/Access/KYCManager.sol
SHA3: 5481e7e586f4e1206982b1bb837685672bfab66c45cdb8f24d9005b02013dca5

File: ./contracts/Access/MasterACL.sol
SHA3: a16cd98424eae0c685d283719f51153c6c58112533c09f97a3f2430e837549c1

File: ./contracts/Access/PrivilegedGroup.sol
SHA3: a16e578d582b619af41547f67348849881fa06d4994ab41339e93ac2b07ab392

File: ./contracts/Access/PrivilegedGroupUpgradeable.sol
SHA3: c83ff1cd2656b231d02d7be794b323cba0e25a8705c4667cc5ab56cece13f010

File: ./contracts/EarlyStage/AnalysisManager.sol
SHA3: 02c0d120d3f2bb79de727eb28adc8a727344fbe695f64db89176ec405d5ea293

File: ./contracts/EarlyStage/CommentManager.sol
SHA3: aea14222065c073c89429e649218ee450092ed02ab5caec5b20b70448a71abe3

File: ./contracts/EarlyStage/CountdownManager.sol
SHA3: d873a3cafade67cf2a801162b0fae2db1e4c92fa6e4829ff3a394f5bef9a71e6

File: ./contracts/EarlyStage/EarlyStageManager.sol
SHA3: 6b5e17c0503a56a7037e6d74fab27564c4c700a1be5640c85106f8f51eba8f1c

File: ./contracts/EarlyStage/EventStoredList.sol
```

```
SHA3: f875d75f00680583a42641229552923d2b7155bdbc581fbba4fa931031bd0e70

File: ./contracts/EarlyStage/ProjectNest.sol
SHA3: 470096ef09629b3edfa2fc2776c11ab1fc512afea51780e6811f563f65df98f0

File: ./contracts/EarlyStage/ProjectNestFactory.sol
SHA3: 7b47726250398fab2be85fc7e285e0251a1d407b254d43b46d6f1537126f7263

File: ./contracts/EarlyStage/UpvoteManager.sol
SHA3: 3df8c8f8c5f6021843609a3be236f2b2b820ea12a802a20271187d5376765322

File: ./contracts/Interfaces/IAnalysisManager.sol
SHA3: 4d8ed69fc9f29a47391cf1face583192f75075e132bdb9645773e6a89cc7ceac

File: ./contracts/Interfaces/ICeTokenDistributionStrategy.sol
SHA3: bb507972c72b564bcf850298b5d1c217402d985bba73bc5eee653670aef019af

File: ./contracts/Interfaces/ICeTokenFactory.sol
SHA3: c6ea5ea7d217224e0d2f8ba8351d202cca64dfe3c7b3cfee41bfd14868b1207f

File: ./contracts/Interfaces/ICollateralToken.sol
SHA3: 8ebba5a1dda1214cf0689e382d060d477f6f96399952d58137e8df8e01f3b802

File: ./contracts/Interfaces/IDiscreteVestingV2.sol
SHA3: a490d2ee8307f9e2f4c94648fef93b71c944ad3b7bafb59cd1cf294ba6362de4

File: ./contracts/Interfaces/IDiscreteVestingV2Factory.sol
SHA3: 106d33368fc53aac74a7ba42bc925b6ab959209a07850bef3852aaa731b94fdb

File: ./contracts/Interfaces/IEarlyStageManager.sol
SHA3: b46a02f144c7df1a358409b0f9eee041284f44e1bed826854989712d1ab4f1de

File: ./contracts/Interfaces/IKYC.sol
SHA3: 36c18fb2033e9bbf119eefa02e01d065411396052ba082749e555a7686f76d89

File: ./contracts/Interfaces/ILinearVestingV2.sol
SHA3: 6fe47b209e512932138a011508563de4e446d82b0da68344e369412f58262f3a

File: ./contracts/Interfaces/ILinearVestingV2Factory.sol
SHA3: adb6e8732147aa53e23e03a1acf7228afee4101864575eca6e010d316baf25d4

File: ./contracts/Interfaces/IMasterACL.sol
SHA3: acc470aa555cdb6206ae7c24ef2bfb48f169975297ddc646e3fe86217872c106

File: ./contracts/Interfaces/IProjectNest.sol
SHA3: 5f845db5e45990bdcacd71d1b62d0f3588f398d41539c920b3c64cbf7aeed417

File: ./contracts/Interfaces/IProjectNestFactory.sol
SHA3: 2ba566e084a72e9dfd2b23b1621ad423abcc5cf01861a0df85a0bec8999e1c80

File: ./contracts/Interfaces/IStakingV2.sol
SHA3: ec3ab8018214964dac46fb2ddc82afb0a3d95a87cf001c92453da5e37e64f5f3

File: ./contracts/Interfaces/IStakingV2Auth.sol
SHA3: dcc074a2a3b9ea80f32ec795e5973b82aec128dba61ca006153d66aec88a6643

File: ./contracts/Interfaces/IStakingV2RewardNotifier.sol
SHA3: 79d7b63df2d346d3fc6ada417a8965d671ed317b1423a5a11aa52244fbad7cbd

File: ./contracts/Interfaces/IVestingV2.sol
```

```
SHA3: b3b6703e553bf4c97af59250f4ba644a0a62b4ac513ecc978ed6ef068464b1ae

File: ./contracts/StakingV2/AntToken.sol
SHA3: 5ffe2f74961557260f82fa1649a8afc5e0ddafb82fd20151330ee6a7eaf38cea

File: ./contracts/StakingV2/StakingV2.sol
SHA3: 419349db75a106394d9d0d8c2684c546e82cf663c0468a6f46ba72de5d571c1c

File: ./contracts/StakingV2/UniversalClaimer.sol
SHA3: 99b69064504c8884cb738a315e729f3765b62fcf2a8838a618b8390382b52d1d

File: ./contracts/Vesting/Vesting.sol
SHA3: 5118e4d14867eb1d23bbf23aa88420871ec5c9d8d0da618e0c9c3d61e7e6fcfc

File: ./contracts/VestingV2/AbstractVestingV2.sol
SHA3: 8cd5601425947353118341dcffd0800388ec9c44085aa699a4db91f556b18a37

File: ./contracts/VestingV2/ceToken.sol
SHA3: 203e9a82b1e6c7a854382e35190623767b42dabe9f62088cd6273a9fb031e94a

File: ./contracts/VestingV2/ceTokenDistributor.sol
SHA3: c6963e39f8369f7dca928cdf891d69faf6e7676d50de15aa6496af5811659cab

File: ./contracts/VestingV2/ceTokenFactory.sol
SHA3: 7e5e8c1c946cd3046e3ecabe22e49173fcfc3c2aedd110cc04a5691db4fcb4ee

File: ./contracts/VestingV2/DiscreteVestingV2.sol
SHA3: dcda075e5971b7c4f64df40f284a4be4c85107baa7caacadebf542cdfe5680a5

File: ./contracts/VestingV2/DiscreteVestingV2Factory.sol
SHA3: f0df2d919a06c73293c5d2ae3ae25935c98f3d81ebe54b64c4f0ee281f0dcc12

File: ./contracts/VestingV2/LinearVestingV2.sol
SHA3: c415cba8753810a283a9a9f89b5832733bc97dde17fc3baf87ce08c84651efdd

File: ./contracts/VestingV2/LinearVestingV2Factory.sol
SHA3: 146564aceecbba131a1507960ccdb1d1a0ca37ccfb708cd9b0232d1dac09e2a2
```

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions.
Medium	Medium-level vulnerabilities are important to fix; however, they cannot lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution.

Executive Summary

The score measurement details can be found in the corresponding section of the scoring methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**.

- Functional and technical requirements are provided.
- Code is followed by NatSpec comments.

Code quality

The total Code Quality score is **10** out of **10**.

- Code follows best practices.

Test coverage

Test coverage of the project is **90%**.

- Deployment and the majority of user interactions are covered with tests.

Security score

As a result of the audit, the code contains **no** issues. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **9.6**.

1
2
3
4
5
6
7
8
9
10
The final score

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
30 September 2022	1	1	0	0
21 October 2022	0	0	0	0

Checked Items

We have audited the Customers' smart contracts for commonly known and more specific vulnerabilities. Here are some items considered:

Item	Type	Description	Status
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	Passed
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Passed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Passed
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	Passed
Access Control & Authorization	CWE-284	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	Not Relevant
Check-Effect-Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	Passed
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	Passed
Delegatecall to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	Passed
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	Passed
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed

Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	Not Relevant
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Passed
Signature Unique Id	SWC-117 SWC-121 SWC-122 EIP-155	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifiers should always be used. All parameters from the signature should be used in signer recovery	Not Relevant
Shadowing State Variable	SWC-119	State variables should not be shadowed.	Passed
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Passed
Calls Only to Trusted Addresses	EEA-Level-2 SWC-126	All external calls should be performed only to trusted addresses.	Passed
Presence of unused variables	SWC-131	The code should not contain unused variables if this is not justified by design.	Passed
EIP standards violation	EIP	EIP standards should not be violated.	Passed
Assets integrity	**Custom**	Funds are protected and cannot be withdrawn without proper permissions.	Passed
User Balances manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Passed
Data Consistency	**Custom**	Smart contract data should be consistent all over the data flow.	Passed
Flashloan Attack	**Custom**	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used.	Not Relevant
Token Supply manipulation	**Custom**	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the customer.	Passed

Gas Limit and Loops	**Custom**	Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	Passed
Style guide violation	**Custom**	Style guides and best practices should be followed.	Passed
Requirements Compliance	**Custom**	The code should be compliant with the requirements provided by the Customer.	Passed
Environment Consistency	**Custom**	The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Passed
Secure Oracles Usage	**Custom**	The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles.	Not Relevant
Tests Coverage	**Custom**	The code should be covered with unit tests. Test coverage should be 100%, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Failed
Stable Imports	**Custom**	The code should not reference draft contracts, that may be changed in the future.	Passed

System Overview

Colony App is a mixed-purpose system with the following contracts:

- MasterACL.sol - manages access control. In addition to managing 2 roles of admin and moderator, the contract is integrated with StakingV2 and KYCManager.
- PrivilegedGroup.sol - simple access control contract. Provides modifiers and functions to set and check privileged accounts.
- PrivilegedGroupUpgradeable.sol - contract adapted to be inherited by upgradeable contracts.
- KYCManager.sol - simple KYC management contract, which allows MasterACL admins to manually change accounts’ KYC compliance.
- AnalysisManager.sol - responsible for project analysis. Allows accounts to submit project analysis, store it, and calculate average values.
- CommentManager.sol - responsible for project comments. Registered accounts can create comments through data URI.
- CountdownManager.sol - manages projects countdown timestamps. The data (data URI) set in this contract is for reference only and is not related to any logic of other early-stage functionalities.
- EarlyStageManager.sol - the main contract which allows for managing and navigating projects through the early stage process.
- EventStoredList.sol - contract, which allows the creation, update, and hide of simple string data.
- ProjectNest.sol - stores and calculates accounts allocations and investments for a specific EarlyStage.
- UpvoteManager.sol - allows accounts to upvote a project.
- AntToken.sol - ERC20 token linked and complementary to RewardingStaking.
- StakingV2.sol - implements an algorithm for multitokens rewards distribution. Adds authorization requirements to the simple staking base with a stake and unstake functionalities, stores account balances, and keeps track of stake total supply and corresponding registered values.
- UniversalClaimer.sol - universal airdrop claimer with support for staking v1 and v2.
- AbstractVestingV2.sol - abstract contract with general implementation for both linear and discrete vestings.
- ceToken.sol - ERC20 token with the functionality of burning tokens for owner accounts.
- ceTokenFactory.sol - contract responsible for deploying new CeToken contract instances.
- ceTokenDistributor.sol - is a contract responsible for deploying new CeToken contract instances. Applies distribution strategy to newly minted CeTokens.

- AbstractVestingV2.sol - an abstract contract with general implementation for both linear and discrete vestings.
- LinearVestingV2.sol - exact linear vesting implementation based on AbstractVestingV2.
- LinearVestingV2Factory.sol - factory contract used for deployment of linear vesting contracts.
- DiscreteVestingV2.sol - exact discrete vesting implementation based of AbstractVestingV2.
- VestingV2Factory.sol - factory contract used for deployment of linear and discrete vesting contracts.

Privileged roles

- The owner of *KYCManager* can set a *MasterACL* contact address.
- The admin of *KYCManager* can set address compliant.
- The owner of *MasterACL* can set admins, set moderators, set KYC managers and set StakingV2 address.
- The owner of *PrivilegedGroup* can update privileged accounts.
- The owner of *PrivilegedGroupUpgradeable* can update privileged accounts.
- The registered user of *AnalysisManager* can submit an analysis for a project.
- The owner of *AnalysisManager* can set early stage manager addresses and master ACL addresses.
- The owner of *CommentManager* can set early access manager address, master ACL and comments per phase limit.
- The registered user of *CommentManager* can create comments.
- The manager of *CommentManager* can hide comments.
- The owner of *CountdownManager* can set an early stage manager and master ACL.
- The admin of *CountdownManager* can emit countdown timestamp, hide and unhide countdown.
- Registered users and KYC compliant of *EarlyAccessManager* can increase allocation and increase investment.
- Registered users of *EarlyAccessManager* can reduce allocation.
- Admin of *EarlyAccessManager* can update project data, update vesting details, update project phase, emit project hidden flag and initialize project nest.
- The owner of *EarlyAccessManager* can set final investment, update vesting data linear, update vesting data discrete, update linear vesting parameters, update discrete vesting parameters, set master ACL, set project nest factory, set analysis manager, set creator cooldown period, update project nest linear vesting factory and update project nest discrete vesting factory.

- The admin of *EventStoredList* can create data events, update emitted data and hide emitted data.
- The owner of *EventStoredList* can update master ACL.
- The owner *ProjectNest* can initialize nest, close nest, increase allocation, reduce allocation, increase investment, penalize allocation, refund stablecoin, set final investment, set linear vesting parameters, set discrete vesting parameters, set linear vesting factory, set discrete vesting factory.
- The owner of *ProjectNestFactory* can set EarlyStageManager,, discrete vesting factories.
- The early stage manager can create project nest.
- The owner of *UpvoteManager* can set MasterACL contract and EarlyStageManager.
- The registered users of *UpvoteManager* can upvote projects.
- The owner of *ColonyGovernanceToken* can initially mint tokens to the receivers and make a snapshot.
- The owner of *AntToken* can set staking address, distribution penalty, redistribution period, and enable or disable transfers.
- The privileged addresses of *AntToken* can set collateral for other addresses, mint, and burn tokens.
- The owner of *StakingV2* can set the staking and unstaking fee, redistribution period, pause and unpause staking, remove rewards, set authorized stake amount and period, set Migrator, Staking, MerkleDistributor contracts, and migration registration expiration period.
- The Ant token of *StakingV2* can unstake users' stakes and change stake ownership.
- The privileged addresses of *StakingV2* can unstake users' stakes.
- The owner of *UniversalClaimer* can register MerkleDistributor
- The owner of *ceToken* can burn tokens and burn them from an address.
- The owner of *CeTokenDistributor* can set shares, dex, and colony addresses.
- The owner of *CeTokenFactory* can set ICeTokenDistributionStrategy.
- The owner of *DiscreteVestingV2Factory* can set CeTokenFactory.
- The owner of *LinearVestingV2Factory* can set CeTokenFactory.

Risks

- System owners and admins can affect many projects' functions and processes.
- In case of admins keys leak, malicious actors will be able to access critical functionality.

Findings

■■■■ Critical

No critical severity issues were found.

■■■ High

No high severity issues were found.

■■ Medium

1. Unoptimized Loops Usage

The contracts use loops without optimization. Array size inside a loop can be cached; state variables should be saved to local memory for any interactions inside a loop.

This will lower Gas taxes.

Paths:
./contracts/EarlyStage/ProjectNestFactory.sol:getProjectNests();
./contracts/StakingV2/StakingV2.sol:getAllRewards();
./contracts/StakingV2/StakingV2.sol:removeReward();
./contracts/StakingV2/StakingV2.sol:updateRewards();
./contracts/StakingV2/UniversalClaimer.sol:_claimAllV1();
./contracts/VestingV2/DiscreteVestingV2.sol:unlockedProjectTokensTotal();
./contracts/VestingV2/DiscreteVestingV2.sol:_updateVestingParameters();
./contracts/VestingV2/DiscreteVestingV2Factory.sol:validateDiscreteVestingParameters();

Recommendation: Cache arrays in a loop, save state variables to local memory, iterate the loop and save changes to the state after the loop finishes.

Status: Fixed (Revised commit: 911229a)

■ Low

2. Floating Pragma

Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Paths: ./contracts/Access/PrivilegedGroup.sol;
./contracts/VestingV2/AbstractVestingV2.sol;
./contracts/VestingV2/ceToken.sol;
./contracts/VestingV2/ceTokenDistributor.sol;
./contracts/VestingV2/ceTokenFactory.sol;
./contracts/VestingV2/DiscreteVestingV2.sol;
./contracts/VestingV2/DiscreteVestingV2Factory.sol;
./contracts/VestingV2/LinearVestingV2.sol;
./contracts/VestingV2/LinearVestingV2Factory.sol;

Recommendation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment.

Status: Fixed (Revised commit: 911229a)

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted to and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, Consultant cannot guarantee the explicit security of the audited smart contracts.