#HashEx
BLOCKCHAIN SECURITY

Oracul Analytics

smart contracts preliminary audit report for internal use only

March 2025

hashex.org

contact@hashex.org

Contents

1. Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and HashEx and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (HashEx) owe no duty of care towards you or any other person, nor does HashEx make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties, or other terms of any kind except as set out in this disclaimer, and HashEx hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, HashEx hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HashEx, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of the use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed. HashEx owns all copyright rights to the text, images, photographs, and other content provided in the following document. When using or sharing partly or in full, third parties must provide a direct link to the original document mentioning the author (hashex.org).

2. Overview

HashEx was commissioned by the Oracul Analytics team to perform an audit of their smart contract. The audit was conducted between 05/05/2025 and 10/03/2025.

The purpose of this audit was to achieve the following:

- Identify potential security issues with smart contracts
- Formally check the logic behind given smart contracts.

Information in this report should be used for understanding the risk exposure of smart contracts, and as a guide to improving the security posture of smart contracts by remediating the issues that were identified.

The code is available at @oracul1/oracul-analytics-sc Gitlab repository and was audited after the commit 8fcd995062f74259d01b6aa1239e4480d1fec30c.

2.1 Summary

Project name	Oracul Analytics
URL	https://oracul.io
Platform	Optimism
Language	Solidity
Centralization level	Medium
Centralization risk	High

2.2 Contracts

Name	Address
OraculAnalytics	0x74d5B4A317fFfd3cBF8Ed94327FF07E2884E84BC
VestingContract	
VestingFactory	0x62E40d9887d083085241993D9196CA92029e6935
MiddlewareContract	0xB55d0a921aa95AB4f882AD4695f32b27Cf085Bb6

3. Project centralization risks

Ca2CR3d Governance functions

The contract owner is responsible for the vesting start timestamps.

The owner provides list of beneficiaries for every vesting group.

The owner must set correct token address.

4. Found issues

Low	4 (67%)
Info	2 (33%)

Ca0. VestingContract

ID	Severity	Title	Status
Ca0I12	Low	Gas optimizations	Acknowledged
Ca0I13	Info	Limited ERC20 token support	Acknowledged

Ca1. VestingFactory

ID	Severity	Title	Status
Ca1I14	Low	Gas optimizations	Acknowledged
Ca1I15	Info	Lack of monitoring tools	Acknowledged

Ca2. MiddlewareContract

ID	Severity	Title	Status
Ca2I16	Low	Gas optimizations	Acknowledged
Ca2I1e	Low	Lack of protection against duplicate vesting calls	Acknowledged

5. Contracts

C9f. OraculAnalytics

Overview

An ERC20 standard token contract. The initial supply is fixed, i.e. there's no active minting functionality after deployment. The token contract is inherited from OpenZeppelin's ERC20 implementation, which is considered the best practice.

No issues were found.

Ca0. VestingContract

Overview

Vesting contract is designed to be deployed by a factory contract, it's initialized with a single ERC20 token address, start timestamp, array of consecutive time periods, and list of beneficiaries. All contract parameters can't be updated after initialization.

Issues

Ca0I12 Gas optimizations

Low Acknowledged

1. Multiple reads from storage in the `claim()` function: `_vesting` structure is used directly and in the `availableToClaim()` function.

2. Multiple reads from storage in the `totalLocked()` function: `_initialTotalLocked` variable.

3. Multiple reads from storage in the `lockedOf()` function: `_initialLocked[account]` variable.

4. Multiple reads from storage in the `_computeUnlocked()` function: `_vesting.startTime`, `_vesting.periods` variables.

5. Multiple reads from storage in the `_initializeBeneficiaries()` function: `_initialTotalLocked` variable.

6. Immutable parameters can be initialized by `Clones.cloneWithImmutableArgs` from OpenZeppelin v5.2 release.

Ca0I13 Limited ERC20 token support

Info Acknowledged

The VestingContract is designed to hold specific ERC20 tokens for a long period of time. This ERC20 token contract must not be a rebasing token that can recalculate balance over time. Additionally, all incoming transfers can't be claimed or unlocked.

Initialization ensures that sum of beneficiaries' amounts is transferred to the contract, but it's never checked actual transfer amount. Tokens with transfer tax should not be used unless vesting contract is excluded from tax.

Ca1. VestingFactory

Overview

Factory contract to create multiple instances of VestingContract by ERC-1167 Minimal Proxy (made with Clones library from OpenZeppelin). Vesting creation is restricted to accounts with the MANAGER_ROLE (provided by AccessControl model from OpenZeppelin).

Issues

Ca1I14 Gas optimizations

Low Acknowledged

1. The VestingFactory contract is not designed to hold any tokens. There's no need to restrict access to the `createVesting()` function. The bytecode could be reduced by removing the AccessControl and related functionality.

Ca1I15 Lack of monitoring tools

Info Acknowledged

The basic version of the AccessControl contract does not allow to get all members of certain roles. The AccessControlEnumerable extension contains additional getters.

Ca2. MiddlewareContract

Overview

Treasury contract to hold the total supply of OraculAnalytics tokens until the owner initiates distribution across destinations with individual vesting schedules.

Vesting schedules are immutably configured for seed, private, public, marketing, liquidity, dao, team, advisors, and ecosystem groups, each consist of list of beneficiaries to be specified by the owner at the moment of vesting start.

The schedules are split by periods of 30 days. The total supply is going to be distributed fully between 9 groups and the total vesting period is not less than 1500 days (50 periods of 30 days for ecosystem group if vesting starts immediately with the token deployment).

Issues

Ca2I16 Gas optimizations

Low Acknowledged

1. Multiple reads from storage in the `setBaseToken()` function: `_baseToken` variable.

Ca2I1e Lack of protection against duplicate vesting calls

Low Acknowledged

The contract allows the owner to initiate vesting for different beneficiary groups by calling functions such as `startSeedVesting()`, `startPrivateVesting()`, and others. However, there is no mechanism to prevent these functions from being called multiple times. If a function is mistakenly executed twice, a duplicate vesting contract will be created, and tokens will be

transferred to it. This could result in insufficient tokens remaining for other vesting schedules, disrupting the vesting process.

Recommendation

Implement a state-tracking mechanism to ensure that each vesting function can only be called once.

6. Conclusion

4 low severity issues were found during the audit. No issues were resolved in the update. The reviewed contracts are highly dependent on the owner's account. See the centralization risks chapter.

This audit includes recommendations on code improvement and the prevention of potential attacks.

Appendix A. Issues' severity classification

- **Critical.** Issues that may cause an unlimited loss of funds or entirely break the contract workflow. Malicious code (including malicious modification of libraries) is also treated as a critical severity issue. These issues must be fixed before deployments or fixed in already running projects as soon as possible.
- **High.** Issues that may lead to a limited loss of funds, break interaction with users, or other contracts under specific conditions. Also, issues in a smart contract, that allow a privileged account the ability to steal or block other users' funds.
- **Medium.** Issues that do not lead to a loss of funds directly, but break the contract logic. May lead to failures in contracts operation.
- **Low.** Issues that are of a non-optimal code character, for instance, gas optimization tips, unused variables, errors in messages.
- **Informational.** Issues that do not impact the contract operation. Usually, informational severity issues are related to code best practices, e.g. style guide.

Appendix B. Issue status description

Resolved. The issue has been completely fixed.

Partially fixed. Parts of the issue have been fixed but the issue is not completely resolved.

Acknowledged. The team has been notified of the issue, no action has been taken.

Open. The issue remains unresolved.

Appendix C. List of examined issue types

- Business logic overview
- Functionality checks
- Following best practices
- Access control and authorization
- Reentrancy attacks
- Front-run attacks
- DoS with (unexpected) revert
- DoS with block gas limit
- Transaction-ordering dependence
- ERC/BEP and other standards violation
- Unchecked math
- Implicit visibility levels
- Excessive gas usage
- Timestamp dependence
- Forcibly sending ether to a contract
- Weak sources of randomness
- Shadowing state variables
- Usage of deprecated code

Appendix D. Centralization risks classification

Centralization level

- **High.** The project owners can manipulate user's funds, lock user's funds on their will (reversible or irreversible), or maliciously update contracts parameters or bytecode.
- **Medium.** The project owners can modify contract's parameters to break some functions of the project contract or contracts, but user's funds remain withdrawable.
- **Low.** The contract is trustless or its governance functions are safe against a malicious owner.

Centralization risk

- **High.** Lost ownership over the project contract or contracts may result in user's losses. Contract's ownership belongs to EOA or EOAs, and their security model is unknown or out of scope.
- **Medium.** Contract's ownership is transferred to a contract with not industry-accepted parameters, or to a contract without an audit. Also includes EOA with a documented security model, which is out of scope.
- **Low.** Contract's ownership is transferred to a well-known or audited contract with industry-accepted parameters.

 contact@hashex.org

 @hashex_manager

 blog.hashex.org

 linkedin

 github

 twitter

#HashEx
BLOCKCHAIN SECURITY