Kinto Token&Vesting Security Analysis

by Pessimistic

This report is public

March 19, 2024

Abstract

In this report, we consider the security of smart contracts of Kinto Token&Vesting project. Our task is to find and describe security issues in the smart contracts of the platform.

Disclaimer

Summary

In this report, we considered the security of Kinto Token&Vesting smart contracts. We described the audit process in the section below.

The initial audit showed one critical issue: Inaccurate timestamp. The audit also revealed two issues of medium severity: Project roles, Lack of transfer allowance. Moreover, one low-severity issue was found. All the tests passed.

After the initial audit, the codebase was updated.

The severity of the critical Inaccurate timestamp issue was changed to medium. The developers fixed the Inaccurate timestamp, Lack of transfer allowance and all the low-severity issues and notes. They also commented on the Project roles issue of medium-severity. The developers increased the number of tests.

The documentation was not provided.

General recommendations

We recommend adding documentation.

Project overview

Project description

For the audit, we were provided with Kinto Token&Vesting project on a public GitHub repository, commit 5d1e8ccc1ee077eef7bdc80e10910bc33a11ca05.

The scope of the audit included:

- **src/tokens/KintoToken.sol**;
- **src/interfaces/IVestingContract.sol**;
- **src/tokens/VestingContract.sol**.

The developers did not provide the documentation for the codebase.

All 343 tests pass successfully. The code coverage is 98.52%.

The total LOC of audited sources is 221.

Codebase update #1

After the initial audit, the codebase was updated, and we were provided with commit ad1f7c6170f2d6140b16306bdabd1b0b704a807f.

The developers fixed or commented on all the medium severity issues. The severity of the critical issue was changed to medium.

The number of tests increased to 399. All of them passed. The code coverage was 97.14%

Audit process

We started the audit on March 7, 2024, and finished on March 11, 2024.

We inspected the materials provided for the audit, then conducted preliminary research and started the review. As we progressed, we reached out to the developers to gather more information about the project and to address any questions that occurred during the review and the research.

We manually analyzed all the contracts within the scope of the audit and checked their logic. Among other, we verified the following properties of the contracts:

- The beneficiary is no longer able to claim the tokens after an early leave;
- The beneficiary can release the vested tokens after a lock period has passed;
- The constant parameters, including a vesting formula, meet the specification;
- There are no possibilities to initiate restricted transfers;
- The inheritance chain is handled correctly when overriding functionality.

We scanned the project with the following tools:

- Static analyzer Slither;
- Our plugin Slitherin with an extended set of rules;
- Semgrep rules for smart contracts. We also sent the results to the developers in the text file.

We ran tests and calculated the code coverage.

In a private report, we combined all the verified issues we found during the manual audit or discovered by automated tools.

We made the recheck #1 on March 19, 2024. We checked whether the previous issues were fixed. Also, we re-ran the tests and calculated the code coverage. Finally, we updated the report.

Manual analysis

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger project security. They can lead to loss of funds or other catastrophic consequences. The contracts should not be deployed before these issues are fixed.

The audit showed no critical issues.

Medium severity issues

Medium severity issues can influence project operation in the current implementation. Bugs, loss of potential income, and other non-critical failures fall into this category, as well as potential problems related to incorrect system management. We highly recommend addressing them.

M01. **Project roles** (commented)

In the context of the **VestingContract**, an `owner` can:

- remove the beneficiary, even if the `vestedAmount` is not zero;
- transfer the vested tokens to any address by invoking `emergencyDistribution`;
- call `earlyLeave` function for any beneficiary and stop them from claiming tokens.

In the context of the **KintoToken** contract, an `owner` can:

- set an arbitrary `_vestingContract`;
- set an arbitrary `_miningContract`;
- enable the token transfers;
- mint tokens.

In the current implementation, the system depends heavily on the owner's role. Thus, there are scenarios that can lead to undesirable consequences for the project and its users, e.g., if the owner's private keys become compromised.

We recommend designing contracts in a trustless manner or implementing proper key management, e.g., setting up a multisig.

> *Comment from the developers: The contract is owned by our Gnosis Safe Multisig with 3 of 5 signers and all those privileged roles will be handed out to on-chain governance as soon as Governance is released at the end of May.*

M02. Lack of transfer allowance (fixed)

Before the governance deployment, token transfers between the users are supposed to be disabled. However, there are specific accounts that should be able to receive and transfer the tokens.

The check at line 94 in the **KintoToken** contract does not include `to != miningContract`, meaning that when transfers are disabled, the mining contract can receive the tokens only if `from == address(0)` or `from == vestingContract`. At this point, it is only possible to receive the tokens from the `vestingContract` by adding the `miningContract` as a beneficiary.

Consider including the `to != miningContract` or `from != owner` in the expression mentioned above to make direct transfers before enabling the token transfers.

The issue has been fixed and is not present in the latest version of the code.

M03. Inaccurate timestamp (fixed)

The `GOVERNANCE_RELEASE_DEADLINE` constant from the **KintoToken** contract is inaccurately assigned to `1714489`, which corresponds to January 20, 1970.

It is inconsistent with the comment in line 36. Also, it allows bypassing the `GovernanceDeadlineNotReached` check upon invoking the `mint` or `enableTokenTransfers` functions, providing an opportunity to mint or enable transfers before an actual governance deployment.

On top of that, the `getSupplyCap` function always misses the `block.timestamp < GOVERNANCE_RELEASE_DEADLINE` check and returns the `MAX_SUPPLY_LAUNCH` value.

The issue has been fixed and is not present in the latest version of the code.

Low severity issues

Low severity issues do not directly affect project operation. However, they might lead to various problems in future versions of the code. We recommend fixing them or explaining why the team has chosen a particular option.

L01. Intrinsic gas optimization (fixed)

There are several ways to optimize the gas cost in the `_update` function of the **KintoToken** contract:

- `!tokenTransfersEnabled` check might be placed as a first expression to bypass all the other checks after enabling the transfers;
- Consider calling the `super._update` after performing necessary checks at lines 94-97 to reduce the gas consumption on failures.

The issues have been fixed and are not present in the latest version of the code.

Notes

N01. Missing events for critical parameters change (fixed)

Consider emitting events in the following functions to improve off-chain integration with the **KintoToken** contract:

- `KintoToken.setVestingContract`;
- `KintoToken.setMiningContract`;
- `KintoToken.enableTokenTransfers`.

The issues have been fixed and are not present in the latest version of the code.

N02. Redundant check (fixed)

The current design of the `KintoToken.mint` function does not allow mint prior to the governance deployment timestamp. To achieve this, the check at line 60 is placed. However, it looks redundant since it is fully covered by the check in line 61. Perhaps it is necessary to revert with a clear error.

The issue has been fixed and is not present in the latest version of the code.

This analysis was performed by Pessimistic:

Daria Korepanova, Senior Security Engineer
Oleg Bobrov, Security Engineer
Rasul Yunusov, Security Engineer
Irina Vikhareva, Project Manager

March 19, 2024