Mokens League Token Audit

16-04-2024

By CoinFabrik

v202311

Executive Summary **3**
Scope **3**
Methodology **3**
Findings **4**
Severity Classification 4
Issues Status 4
Critical Severity Issues 5
High Severity Issues 5
Medium Severity Issues 5
Minor Severity Issues 5
Enhancements 5
Other Considerations **5**
Centralization 5
Upgrades 6
Privileged Roles 6
MokensLeague.sol 6
Changelog **6**

Executive Summary

CoinFabrik was asked to audit the contracts for the Mokens Leage Token project. The audit is based on the commit `af8c8332c054f4f84db0cf191b6cdf27c88047ef`. The project consists of a customized ERC20 Ethereum token.

During this audit we found no critical issues, no medium issues and no minor issues.

No fixes were required by the development team.

Scope

The audited files are from the git repository located at https://github.com/wakeuplabs/mokens-tge. The audit is based on the commit `<hash>`.

The scope for this audit includes and is limited to the following 2 files:

- `contracts/MokensLeague.sol`: ERC20 Token
- `contracts/IMokensLeagueErrors`: Errors definition

No other files in this repository were audited. Its dependencies are assumed to work according to their documentation. Also, no tests were reviewed for this audit.

Methodology

CoinFabrik was provided with the source code. Our auditors spent one day auditing the source code provided, which includes understanding the context of use, analyzing the boundaries of the expected behavior of each contract and function, understanding the implementation by the development team (including dependencies beyond the scope to be audited) and identifying possible situations in which the code allows the caller to reach a state that exposes some vulnerability. Without being limited to them, the audit process included the following analyses.

- Arithmetic errors
- Outdated version of Solidity compiler
- Race conditions
- Reentrancy attacks
- Misuse of block timestamps
- Denial of service attacks
- Excessive gas usage
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling

- Insufficient validation of the input parameters
- Incorrect handling of cryptographic signatures
- Centralization and upgradeability

Findings

In the following table we summarize the security issues we found in this audit. The severity classification criteria and the status meaning are explained below. This table does not include the enhancements we suggest to implement, which are described in a specific section after the security issues.

No issues were identified in this code.

Severity Classification

Security risks are classified as follows:

- **Critical:** These are issues that we manage to exploit. They compromise the system seriously. Blocking bugs are also included in this category. They must be fixed **immediately**.
- **High:** These refer to a vulnerability that, if exploited, could have a substantial impact, but requires a more extensive setup or effort compared to critical issues. These pose a significant risk and **demand immediate attention**.
- **Medium:** These are potentially exploitable issues. Even though we did not manage to exploit them or their impact is not clear, they might represent a security risk in the near future. We suggest fixing them **as soon as possible**.
- **Minor:** These issues represent problems that are relatively small or difficult to take advantage of, but might be exploited in combination with other issues. These kinds of issues do not block deployments in production environments. They should be taken into account and be fixed **when possible**.

Issues Status

An issue detected by this audit has one of the following statuses:

- **Unresolved**: The issue has not been resolved.
- **Acknowledged**: The issue remains in the code, but is a result of an intentional decision. The reported risk is accepted by the development team.
- **Resolved**: Adjusted program implementation to eliminate the risk.

- **Partially resolved**: Adjusted program implementation to eliminate part of the risk. The other part remains in the code, but is a result of an intentional decision.
- **Mitigated**: Implemented actions to minimize the impact or likelihood of the risk.

Critical Severity Issues

No issues found.

High Severity Issues

No issues found.

Medium Severity Issues

No issues found.

Minor Severity Issues

No issues found.

Enhancements

These items do not represent a security risk. They are best practices that we suggest implementing.

No enhancements are required in this project.

Other Considerations

The considerations stated in this section are not right or wrong. We do not suggest any action to fix them. But we consider that they may be of interest to other stakeholders of the project, including users of the audited contracts, token holders or project investors.

Centralization

The contract is pausable by the owner, and also the initial holder account is the only one that can mint tokens, and just at the creation of the token.

Upgrades

No upgrade mechanism is provided for this contract.

Privileged Roles

These are the privileged roles that we identified on each of the audited contracts.

MokensLeague.sol

Owner

The contract owner can pause and unpause all transfers on this token.

Changelog

- 16-04-2024 – Initial report based on commit `af8c8332c054f4f84db0cf191b6cdf27c88047ef`.

Disclaimer: This audit report is not a security warranty, investment advice, or an approval of the Mokens League Token project since CoinFabrik has not reviewed its platform. Moreover, it does not provide a smart contract code faultlessness guarantee.