AudiTBlock

RollBitPepe ($ROB)

v0.8.19+commit.7dd6d404

v0.8.13

Low-Risk	Medium-Risk	High-Risk
Low-risk code	Medium-risk code	High-risk code

RollBitPepe

0x617B678DDbde93Fd165FBe7Be29CE1485b1f0baA

Executive Summary

Project Name RollBitPepe

Overview The RollBitPepe ecosystem is designed to provide players with an immersive and rewarding experience. The ecosystem is constantly evolving, and new features and benefits are being added all the time

Method Manual Review, Functional Testing, Automated Testing etc.

Scope of Audit The scope of this audit was to analyze the contract codebase for quality, security, and correctness.

High Medium

Low Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	0	0

Smart Contract Weakness Classification (SWC) Vulnerabilities for Attacks

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- Dangerous strict equalities
- Tautology or contradiction
- Missing Zero Address Validation
- Return values of low-level calls
- Revert/require functions
- Private modifier
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

Types of Severities

High

A high-severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Techniques and Methods

The overall quality of code.

- Use of best practices.
- Code documentation and comments match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.
- implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrance and other vulnerabilities.

The following techniques, methods, and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, and their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms Used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Phase 1

Project - RollBitPepe

High Severity Issues

No issues found

Medium Severity Issues

No issues found

Low Severity Issues

1. Ownership Methods (manipulate ownership)

```
        transferOwnership(initialOwner);
    }
    function mint(address account, uint256 amount) external onlyOwner {
        _mint(account, amount);
    }


    function changeOwner(address newOwner) external onlyOwner {
        transferOwnership(newOwner);
```

Description

Our auditor found this issue manually. It is not a bug or vulnerability. We are simply acknowledging that the contract already uses OpenZeppelin's ownership support contract, and the contractor again defines the initial ownership slot. This may cause damage or danger.

Recommendation

It is important to note that not define the ownership with external calls. You can use the Openzeppelin ownership contract. Which you have already used! However, staying away from calling ownership with external calls. Just use a modifier for that. Using modifiers is good practice.

Status

Acknowledged

Informational Severity Issues

No issues found

Phase 2

```
RollBitPepe.constructor(string,string,address,uint256).name
(contracts/RollBitPepe.sol#9) shadows:
        - ERC20.name()
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)
(function)
        - IERC20Metadata.name()
(node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metada
ta.sol#17) (function)
RollBitPepe.constructor(string,string,address,uint256).symbol
(contracts/RollBitPepe.sol#10) shadows:
        - ERC20.symbol()
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
(function)
        - IERC20Metadata.symbol()
(node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metada
ta.sol#22) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#local-variable-shadowing
```

```
Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.13']
        - ^0.8.0
(node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4)
        - ^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnabl
e.sol#4)
        - ^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metada
ta.sol#4)
        - ^0.8.0
(node_modules/@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.13 (contracts/RollBitPepe.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#different-pragma-directives-are-used
```

```
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allows old
versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old
versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol#
4) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol
#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
allows old versions
Pragma version^0.8.13 (contracts/RollBitPepe.sol#2) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#incorrect-versions-of-solidity
```

Functional Testing

Some of the tests performed are mentioned below:

- ✔ Should revert when non-owner calls the changedOwner
- ✔ Should be applied initial supply
- ✔ Should revert when none-owner calls transform-from
- ✔ Should work mint method correctly
- ✔ Should mint to any address

Closing Summary

In this report, we have considered the security of RollBitPepe. We performed our audit according to the procedure described above.

Several issues were identified during the audit process, and their severity levels have been classified. Recommendations and best practices have also been provided to enhance code quality and security posture. The team has acknowledged all identified issues.

Disclaimer

AuditBlock

AuditBlock is a blockchain security company that provides professional services and solutions for securing blockchain projects. They specialize in smart contract audits on various blockchains and offer a range of services

300+
Audits Completed

$3B
Secured

300K
Lines of Code Audited

Contact For Audit

auditblock@gmail.com

Audit Time,
13 November 2023

Audit Report of RollBitPepe

auditblock@gmail.com

Auditblock