QuillAudits

Audit Report
November, 2022

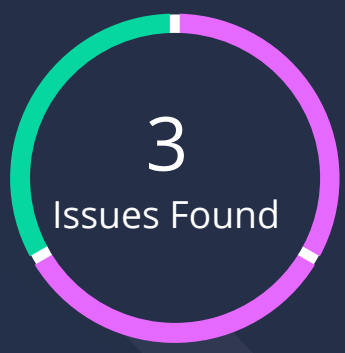
For

finblox

Table of Content

Executive Summary

Project Name	FBXToken
Timeline	FBXToken is ERC20 token with 10B totalsupply, it inherits ERC20 functionalities from OpenZeppelin ERC20 implementation.
Method	November 1,2022 to November 30,2022
Scope of Audit	Manual Review, Functional Testing, Automated Testing etc. The scope of this audit was to analyze Finbloxapp's FBXToken Contract code for quality, security, and correctness.
Fixed In	Zip file was provided by Finbloxapp team



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	1	2

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.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- 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
- Return values of low-level calls
- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

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

Structural Analysis

In this step, we have analysed 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 analysed, 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 behaviour 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.

Manual Testing

A. Contract - FBXToken.sol

High Severity Issues

No issues found

Medium Severity Issues

No issues found

Low Severity Issues

A.1 Unlocked pragma (pragma solidity ^0.8.4)

Description

Contract is using floating pragma (pragma solidity ^0.8.4) Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly. Using floating pragma does not ensure that the contracts will be deployed with the same version. It is possible that the most recent compiler version get selected while deploying contract which has higher chances of having bugs in it.

Remediation

Remove floating pragma and use specific compiler version with which contracts have been tested.

Status

Resolved

FinBlox team's comment: Pragma is locked to solidity version 0.8.9

Informational Issues

A.2 Total supply can be exceeded by deployer by mistake

Description

Constructor takes initialBalance as total supply. the initialBalance can be less or more than 10 billion, this difference doesn't directly affect this smart contract's functionality but it can affect other calculations that are being performed for finding percentage on distribution and allocation.

Remediation

Extra care needs to be taken while specifying total supply while deploying the FBXToken contract.
initialBalance can be hardcoded in smart contract constructor instead of manually entering it while deploying.

Status

Resolved

FinBlox team's comment: Initial supply is hard-coded into the contract to avoid human-mistakes.

A. 3 Check decimals while entering initial supply

Description

The constructor takes initialBalance as argument, and msg.sender gets initialBalance amount of tokens. while entering initial supply it is required to take care of token decimals, token decimals should be added to the entered amount e.g for 10000000000 tokens, 10000000000 *(10**18) will give a token value with 18 decimals which can be then used as initialBalance while deploying the contract.

Remediation

Care needs to be taken while entering initialBalance, the risk of mistake can be minimized by hard coding tokensupply value directly while minting tokens in constructor.

Status

Resolved

FinBlox team's comment: Decimal number is standard, taken from decimals() function

Functional Testing

FBXToken.sol

- ✓ Contract mints initialBalance amount of tokens deployer address
- ✓ Should be able to transfer tokens to other address
- ✓ Should be able to approve tokens
- ✓ Should be able to tranferFrom approved tokens
- ✓ Should be able to increase and decrease allowance
- ✓ Reverts if spender tries to transfer tokens using tranferFrom more than the allowance
- ✓ Reverts if user tries to tranfer tokens more than the amount he holds

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

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

Some issues of Low and informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture. At the end, FinBloxapp team Resolved all Issues

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the Finbloxapp Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Finbloxapp Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

700+
Audits Completed

$15B
Secured

700K
Lines of Code Audited

Follow Our Journey

Audit Report
November, 2022

For

finblox

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com