

blockbank

SMART CONTRACT AUDIT

ZOKYO.

June 8th, 2021 | v. 1.0

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.



TECHNICAL SUMMARY

This document outlines the overall security of the BlockBank token (Binance Smart Chain) smart contracts, evaluated by Zokyo's Blockchain Security team.

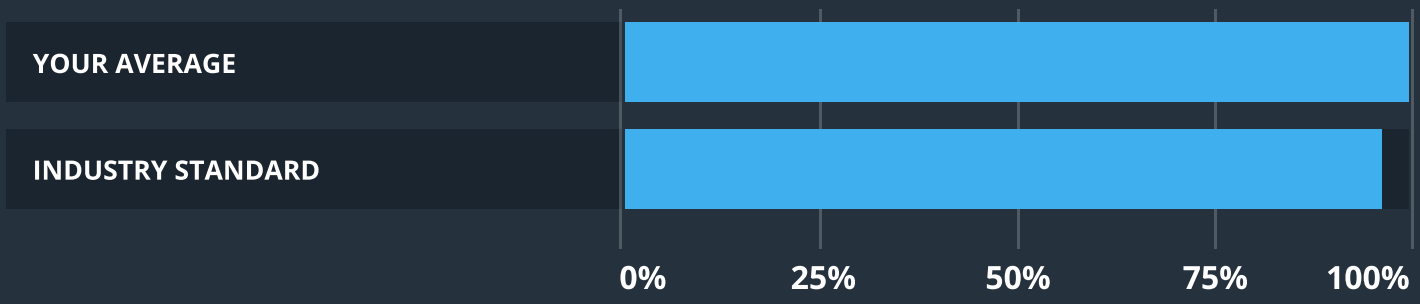
The scope of this audit was to analyze and document the BlockBank token (Binance Smart Chain) smart contract codebase for quality, security, and correctness.

Contract Status



There were no critical issues found during the audit.

Testable Code



The testable code is 100%, which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Zokyo recommend that the BlockBank team put in place a bug bounty program to encourage further and active analysis of the smart contract.

TABLE OF CONTENTS

Auditing Strategy and Techniques Applied 3

Executive Summary 4

Structure and Organization of Document 5

Complete Analysis 6

Code Coverage and Test Results for all files 11

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the BlockBank deployed token (Binance Smart Chain).

Contract address:
https://bscscan.com/address/0xf4b5470523ccd314c6b9da041076e7d79e0df267

Throughout the review process, care was taken to ensure that the token contract:

- Implements and adheres to existing Token standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of gas, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of BlockBank smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1	Due diligence in assessing the overall code quality of the codebase.	3	Testing contract logic against common and uncommon attack vectors.
2	Cross-comparison with other, similar smart contracts by industry leaders.	4	Thorough, manual review of the codebase, line-by-line.

EXECUTIVE SUMMARY

There were no critical issues found during the audit. All the mentioned findings may have an effect only in case of specific conditions performed by the contract owner.

The findings during the audit have a slight impact on contract code style and performance and only may have impact during further development.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

MEDIUM | UNRESOLVED

Max amount may exceed totalSupply

Although these functions can be called only by an authorized person, the total of maxAmounts should be calculated in order not to exceed totalSupply or balance of _lgePairAddress .
Also notice, that max amount for each round is a limit for each recipient, but not a total limit for a whole round. Consider adding more checks.

Recommendation:
Add an internal function which calculates sum of max amounts and reverts if total is exceeded. Call this function in both createLGEWhitelist and modifyLGEWhitelist functions after setting new values to amountsMax. Or confirm the functionality and calculations.

LOW | UNRESOLVED

Not restricted round start

LGEWhitelisted.sol, line 145, _applyLGEWhiteList function
For now anyone can start the purchases rounds (send tokens to _lgePairAddress, thus triggering _applyLGEWhiteList).

Recommendation:
Confirm the functionality or consider restricting for owner/whitelister only.

LOW | UNRESOLVED

Parameters lack zero check

BlockBank.sol, line 16

Function parameters recipient and amount should be checked for misleading transactions prevention (e.g. in case of the frontend validations failure).

Recommendation:
Add “require” statements to check that parameters are not zeros before executing the function.

LOW | UNRESOLVED

Address lacks zero-check

LGEWhitelisted.sol, line 61
Missing “require” statement for comparison “pairAddress” function parameter with zero address. It is needed to prevent misleading transactions or a transaction with incorrect arguments (e.g. because of frontend validation failure).

Recommendation:
Add “require” statement before executing function.

INFO | UNRESOLVED

Use storage pointer

LGEWhitelisted.sol, line 86 “modifyLGEWhitelist” function
There are too many usages for _lgeWhitelistRounds[index]. In order to provide gas savings during the operation, consider usage of storage pointer.

Recommendation:
Use storage pointer instead.

INFO | UNRESOLVED

Missing zero address check

LGEWhitelisted.sol, line 97, modifyLGEWhitelist function
Missing checking addresses from “addresses” array parameter.

Recommendation:
Add “require” statement in for-loop and check every address in the array.

INFO | UNRESOLVED

Variables names too similar.

BlockBank.sol, line 10
Constant TOTAL_SUPPLY is too similar to BEP20._totalSupply and may lead to misunderstanding during development.

Recommendation:
Consider using a different name, i.e SUPPLY_CAP.

INFO | UNRESOLVED

Lacks TOTAL_SUPPLY check in mint function

BEP20.sol, line 213
Mint function lacks check, that _totalSupply + amount will not exceed TOTAL_SUPPLY. Although this function can be called by owner only, this check should be provided.

Recommendation:
Consider creating an override method mint in BlockBand.sol, which will check that _totalSupply + amount does not exceed TOTAL_SUPPLY.

	BlockBank	LGEWhitelisted	BEP20
Re-entrancy	Pass	Pass	Pass
Access Management Hierarchy	Pass	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass	Pass
Unexpected Ether	Pass	Pass	Pass
Delegatecall	Pass	Pass	Pass
Default Public Visibility	Pass	Pass	Pass
Hidden Malicious Code	Pass	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass	Pass
External Contract Referencing	Pass	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass	Pass
Unchecked CALL Return Values	Pass	Pass	Pass
Race Conditions / Front Running	Pass	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass	Pass
Floating Points and Precision	Pass	Pass	Pass
Tx.Origin Authentication	Pass	Pass	Pass
Signatures Replay	Pass	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo team

As part of our work assisting BlockBank in verifying the correctness of their contract code, our team was responsible for writing integration tests using Truffle testing framework.

Tests were based on the functionality of the code, as well as review of the BlockBank contract requirements for details about issuance amounts and how the system handles these.

```
Testing LGEWhitelisted
  Test whitelister transfer functions
    √ Should succesfully transfer and renounce whitelister role (2427ms)
    √ Should not transfer whitelister to zero address (488ms)
    √ Should not let not whitelister call functions (644ms)
  Test create and modify whitelist
    √ Should succesfully create new whitelist (1547ms)
    √ Should not create whitelist if provided params are wrong (836ms)
    √ Should succesfully modify round (1993ms)
    √ Should not modify round if index provided > whitelist rounds length (1227ms)
    √ Should not modify round if max amount equals zero (1215ms)
    √ Should not set same duration and amountMax (1315ms)
  Test apply whitelist
    √ Should not apply whitelist if it is not created (391ms)
    √ Should start whitelist rounds and let whitelisted buyer buy (4922ms)
    √ Should not apply to whitelist if it is not started yet (1668ms)
    √ Should not send tokens to unwhitelisted buyer (1791ms)
    √ Should not send more tokens, than amount max (3919ms)
```

```
  14 passing (41s)

----------------------|----------|----------|----------|----------|----------------|
File                  |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
----------------------|----------|----------|----------|----------|----------------|
 contracts\           |      100 |      100 |      100 |      100 |                |
  BlockBank.sol       |      100 |      100 |      100 |      100 |                |
  LGEWhitelisted.sol  |      100 |      100 |      100 |      100 |                |
----------------------|----------|----------|----------|----------|----------------|
All files             |      100 |      100 |      100 |      100 |                |
----------------------|----------|----------|----------|----------|----------------|
```

We are grateful to have been given the opportunity to work with the BlockBank team.

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.

Zokyo's Security Team recommends that the BlockBank team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.