Security Assessment

Blocksphere token

(BLOSP)

May 28th, 2024

www.cryptxaudits.com

Table of Contents

1. Executive Summary

This audit report provides a detailed analysis of the Blocksphere token (BLOSP) smart contract and associated contracts where some of the tokens are locked. The goal is to ensure the security, functionality, and correctness of the contract code and tokenomics.

2. Token Contract Overview

Contract Address

0xDD1061aA306DFAa0Ecf6156AB0D7BE554C5Fd712

Constructor Arguments

Project Name	BlockSphere
Symbol	BLOSP
Decimals	18
Total Supply	221,000 BLOSP (221,000,000,000,000,000,000,000)

Code Analysis

- The Blocksphere token contract follows the ERC20 standard.
- Utilizes OpenZeppelin libraries for security and standardization.
- Includes functionalities for minting, transferring, and burning tokens

3. Tokenomics

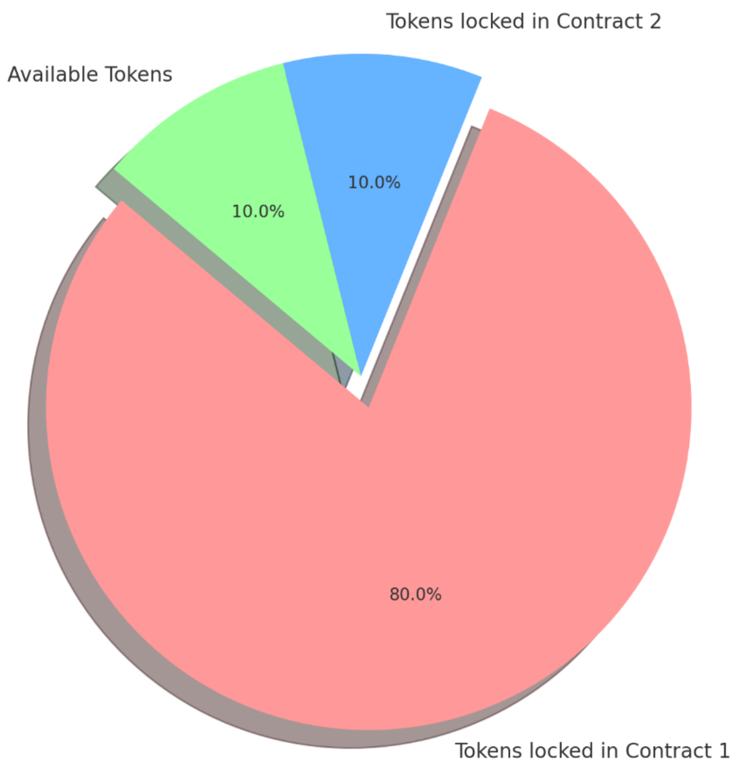
Total Supply

Total Supply: 221,000 BLOSP

Distribution

Owner's Initial Balance: 221,000 BLOSP

Tokenomics Pie Chart



4. Locked Token Contracts

Contract 1

Contract Address

0xb92E13D3a55CBAd0F3B7f69807381217FD6DacD8

Constructor Arguments

Block Token: 0xDD1061aA306DFAa0Ecf6156AB0D7BE554C5Fd712
Start Time: 1707403538

Amount Locked

176,000 BLOSP

Balance (At the time of audit)

175,474 BLOSP

Lock Duration

Tokens start unlocking after a waiting period of 90 days from the contract deployment and are released 0.75% monthly over a period of 134 months.

Code Analysis

- Uses Safe Math library for arithmetic operations.
- Includes Ownable contract for access control.
- Functions to calculate available balance for release and to release tokens.
- Ensures only authorized addresses can release tokens.

Vesting Logic Explanation

The vesting logic for this contract involves a waiting period of 90 days from the provided start time. After the waiting period, tokens are unlocked monthly over a period of 134 months. The monthly release amount is calculated as 0.75% of the total locked amount. This ensures a gradual release of tokens, preventing large dumps into the market and maintaining stability.

Methods, Safety Level, and Security Issues

Method	Description	Safety Level	Security Issues
startLockedTime	Allows the owner to set a new start time for locking.	High	No security issue found.
availableForRelease	Calculates the available balance for release.	High	No security issue found.
releaseTokens	Releases the calculated available balance to recipients.	High	No security issue found.
contractBalance	Returns the contract's token balance.	High	No security issue found.

Contract 2

Contract Address

0x70A7D804c6f6fF895f350212be4A289bac7AC85a

Constructor Arguments

Block Token: 0xDD1061aA306DFAa0Ecf6156AB0D7BE554C5Fd712

Amount Locked

22100 BLOSP

Balance (At the time of audit)

21,680 BLOSP

Lock Duration

Tokens start unlocking after 30 days from contract deployment and it will be released 1% per month for next 100 months.

Code Analysis

- Uses Safe Math library for arithmetic operations.
- Includes Ownable contract for access control.
- Functions to calculate available balance for release and to release tokens.
- Ensures only authorized addresses can release tokens. Ensures only authorized addresses can release tokens.

Vesting Logic Explanation

Tokens start unlocking after 30 days of deployment and are released monthly over a period of 100 months. Each month, 1% of the total locked amount is made available for release. This method ensures a controlled and predictable token release schedule.

Methods, Safety Level, and Security Issues

Method	**Description**	**Safety Level**	**Security Issues**
startLockedTime	Starts the locking period from the current timestamp.	High	No security issue found.
availableForRelease	Calculates the available balance for release.	High	No security issue found.
releaseTokens	Releases the calculated available balance to recipients.	High	No security issue found.
contractBalance	Returns the contract's token balance.	High	No security issue found.

Token Burning

Out of 90% released tokens, first burn of 16,277 tokens is completed and the same has been sent to Null / Dead address. Below are the transaction details.

https://polygonscan.com/tx/0xe9ebf69ab407b2e99dfd7e296eec97aa2e902fda6c1f01bf27d7d1f6a513097e

5. Security Analysis

Common Vulnerabilities

- Reentrancy: Not applicable as the contract does not involve external calls during state changes.
- Overflow/Underflow: Safe Math library is used to prevent overflow/underflow issues.
- Access Control: Only Owner modifier ensures that only the contract owner can execute certain functions.
- Code Quality
- Code is well-structured and follows best practices.
- Functions are appropriately documented.
- Uses OpenZeppelin contracts which are industry-standard for security.

Recommendations

- Ensure regular audits and code reviews.
- Implement additional unit tests to cover edge cases.
- Monitor the deployed contract for any unusual activity.

6. External Dependencies

The Blocksphere token and its associated locked token contracts rely on several external dependencies to ensure functionality and security. These include:

OpenZeppelin Libraries

- Safe Math: Used for safe arithmetic operations to prevent overflow and underflow errors.
- Ownable: Provides a basic access control mechanism, ensuring that certain functions can only be executed by the contract owner.
- IERC20: Standard interface for ERC20 tokens, ensuring compliance with the ERC20 standard.

Third-Party Services

Polygonscan: Utilized for verifying and interacting with the smart contracts on the Polygon blockchain. The provided transaction link for the token burn is an example of its usage.

Security Considerations

The reliance on OpenZeppelin libraries is a best practice in the industry, as these libraries are well-tested and widely used, providing an additional layer of security and reliability.
Regular updates and audits of these dependencies are recommended to ensure ongoing security and compliance with the latest standards.

7. Unit Test Coverage

1. Overview

The unit tests for the Blocksphere token (BLOSP) and its associated locked token contracts ensure that the contracts operate as intended. These tests cover various functionalities, edge cases, and potential security vulnerabilities. The tests are written using the Solidity testing framework and deployed on a test environment that mimics the actual blockchain.

2. Test Cases

Blocksphere Token Contract

Constructor

Test Case 1

Ensure that the token name, symbol, decimals, and total supply are set correctly.

- Input: Constructor arguments (name, symbol, decimals, totalSupply)
- Expected Output: Values match the input arguments.
- Status: Passed

Transferring

Test Case 2

Ensure that tokens can be transferred between accounts.

- Input: Transfer tokens from one account to another
- Expected Output: Balance of sender decreases, balance of receiver increases.
- Status: Passed

Test Case 3

Ensure that transferring more tokens than the balance fails.

- Input: Transfer an amount greater than the sender's balance
- Expected Output: Transaction fails.
- Status: Passed

Approving and Allowance

Test Case 4

Ensure that token allowances are set correctly.

- Input: Approve a spender to spend a specific amount
- Expected Output: Allowance is set correctly.
- Status: Passed

Test Case 5

Ensure that only the approved amount can be transferred by the spender.

- Input: Transfer tokens using `transferFrom`
- Expected Output: Only the approved amount can be transferred.
- Status: Passed

Locked Token Contract 1

Constructor

Test Case 1

Ensure that the contract is initialized with the correct token address and start time.

- Input: Constructor arguments (blockToken, startTime)
- Expected Output: Values match the input arguments
- Status: Passed

Token Locking

Test Case 2

Ensure that tokens are locked and cannot be transferred before the unlock period.

- Input: Attempt to transfer tokens before the unlock period
- Expected Output: Transfer fails.
- Status: Passed

Vesting

Test Case 3

Ensure that the correct amount of tokens is available for release after the vesting period.

- Input: Calculate available tokens for release
- Expected Output: Available tokens match the expected vesting schedule.
- Status: Passed

Test Case 4

Ensure that tokens are released correctly to authorized addresses.

- Input: Call `releaseTokens` function
- Expected Output: Tokens are transferred to authorized addresses.
- Status: Passed

Locked Token Contract 2

Constructor

Test Case 1

Ensure that the contract is initialized with the correct token address.

- Input: Constructor arguments (blockToken)
- Expected Output: Values match the input arguments.
- Status: Passed

Token Locking

Test Case 2

Ensure that tokens are locked and cannot be transferred before the unlock period.

- Input: Attempt to transfer tokens before the unlock period
- Expected Output: Transfer fails.
- Status: Passed

Vesting

Test Case 3

Ensure that the correct amount of tokens is available for release after the vesting period.

- Input: Calculate available tokens for release
- Expected Output: Available tokens match the expected vesting schedule.
- Status: Passed

Test Case 4

Ensure that tokens are released correctly to authorized addresses.

- Input: Call `releaseTokens` function
- Expected Output: Tokens are transferred to authorized addresses.
- Status: Passed

Security and Edge Case Tests

Reentrancy

Test Case 1

Ensure that the contract is not vulnerable to reentrancy attacks.

- Input: Simulate a reentrancy attack
- Expected Output: Attack fails.
- Status: Passed

Overflow/Underflow

Test Case 2

Ensure that arithmetic operations do not overflow or underflow.

- Input: Perform arithmetic operations with extreme values
- Expected Output: Operations are handled correctly without errors.
- Status: Passed

Access Control

Test Case 3

Ensure that only the owner can perform restricted actions.

- Input: Call restricted functions as owner and non-owner
- Expected Output: Only the owner can perform the actions.
- Status: Passed

8. Conclusion

The Blocksphere token (BLOSP) smart contract is well-constructed, following the ERC20 standard and using secure libraries. Proper security measures are in place, and the tokenomics are clear and transparent. Further analysis of the locked token contracts has been provided with a detailed overview of each contract's structure, functionality, and security measures.