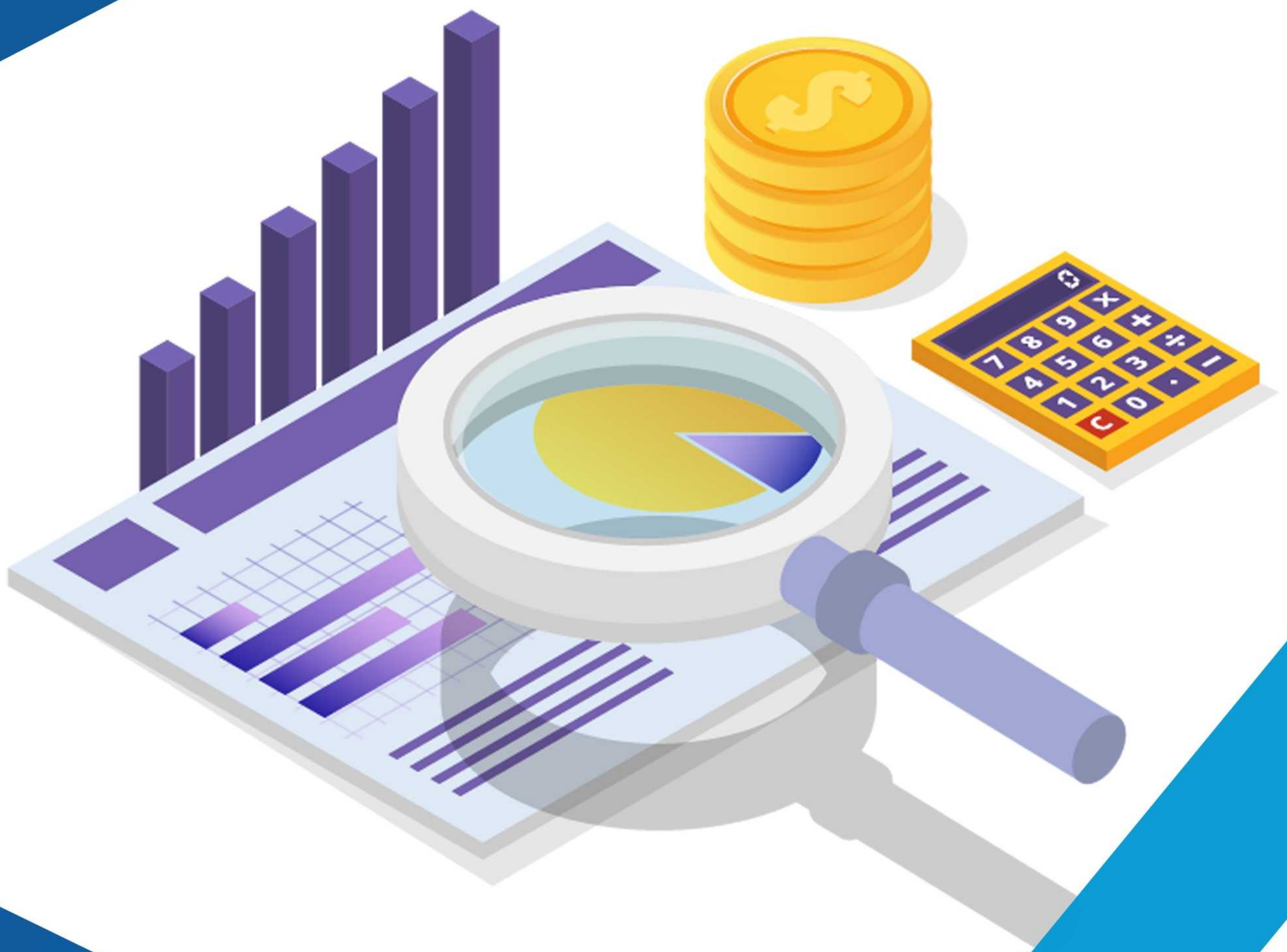
QuillAudits



Audit Report
April, 2021

b-cube.ai

Contents

Overview of Passive Income Bot

B-Cube ICO

The Ultimate All-in-one AI-driven Platform for Crypto Traders

The BCUBE token is the fuel of the ecosystem for gaining access to free trading signals, bots, education courses, and various other benefits by staking the required number of BCUBE tokens.
It can also be used as a means of payment with a discount for bots in addition to the free bots offered by the staking benefits. The platform will also be accessible via payment in FIAT currencies, but that payment made in FIAT currencies will be immediately converted to BCUBE tokens by the team.

Token Name: BCUBE
Tokens available for public sale: 15.000.000 BCUBE
Pre-ICO (Private Sale): 8m
ICO (Public Sale): 7m
Accepted currencies: ETH, USDT

Scope of Audit

The scope of this audit was to analyse B-cube's BCubePublicSale and PublicSaleTreasury smart contracts codebase for quality, security, and correctness.

BCubePublicSale.sol -
https://github.com/b-cube-ai/b-cube-ico/blob/feat/public-sale/contracts/BCubePublicSale.sol

Commit: 170a30447b5f9ee890ade09079d89c40f85091a1

PublicSaleTreasury.sol -
https://github.com/b-cube-ai/b-cube-ico/blob/feat/public-sale/contracts/PublicSaleTreasury.sol

Commit: 79dc0fd60d867e0a564eb51b0ecc89d6f615daf3

Check Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level
- Address hardcoded
- Using delete for arrays
- Integer overflow/underflow
- Locked money
- Private modifier
- Revert/require functions
- Using var
- Visibility
- Using blockhash
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

Techniques and Methods

Throughout the audit of Passive Income Bot smart contracts 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 intended behaviour mentioned in whitepaper.

- 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 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 security of smart contracts.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of 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 possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Mythril, Slither, SmartCheck, Solhint, Surya

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

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 to be fixed before moving to a live environment.

Medium level severity issues

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 level severity issues

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 four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

	High	Medium	Low	Informational
Open	0	0	2	11
Closed	0	1	0	7

Functional Testing

1. Deployer should be a whitelistAdmin? PASS
2. WhitelistAdmin shouldn't be whitelisted? PASS
3. Deployer should be a pauser? PASS
4. Non-Whitelisted Admin should not call setAdmin? PASS
5. Non-Whitelisted Admin should not call addWhitelisted? PASS
6. Non-Whitelisted Admin should not call addWhitelistedAdmin? PASS
7. Non-Whitelisted Admin should not call extendClosingTime? PASS
8. Non-Whitelisted Admin should not call removeWhitelisted? PASS
9. Non Pauser should not call addPauser? PASS
10. Pasuer should call addPauser? PASS
11. Whitelisted Admin should call setAdmin? PASS
12. Calling Whitelisted Admin should be removed from whitelistAdmin role after calling setAdmin? PASS
13. Calling Pauser should be removed from pauser role after calling setAdmin? PASS
14. Non-Whitelisted Admin should not call setETHPriceFeed? PASS
15. Non-Whitelisted Admin should not call setUSDTInstance? PASS
16. Non-Whitelisted Admin should not call setUSDTPriceFeed? PASS
17. Whitelisted Admin should call setETHPriceFeed? PASS
18. Whitelisted Admin should call setUSDTInstance? PASS
19. Whitelisted Admin should call setUSDTPriceFeed? PASS
20. Non-Whitelisted should not call buyBcubeUsingEth? PASS
21. Whitelisted should call buyBcubeUsingEth? PASS
22. Pauser should be able to pause the sale? PASS
23. Pauser should be able to unpause the sale? PASS
24. Non-Pauser should not be able to Pause the sale? PASS
25. Non-Pauser should not be able to Unpause the sale? PASS
26. BCubePreICO should be allotted if the stage is 1 that is netsoldcube 8,000,000e18, BCubeICO should be allotted otherwise? PASS
27. OnlyWhiteListAdmin should change listing time? PASS
28. Listingtime should not be changed after listing? PASS
29. publicSaleShareWithdraw should be callable only afterlisting? PASS

Tried and Tested 29 test cases. PASSED: 29 FAILED: 0

Issues Found – Code Review / Manual Testing

High severity issues

None.

Medium severity issues

BCubePublicSale.sol

1. **Oracle Manipulation Can Lead to Integer Underflow**

   Functions **fetchETHPrice()** [#L285-288] and **fetchUSDTPrice()** [#291-295] are returning signed int256 values. The values will be used as ethPrice[#L199] and usdtPrice[#216]. They are not properly typecasted. In the case of Oracle Manipulation Attack, if the functions return a negative value, it will result in to Integer Underflow.

   Recommendation

   Use SafeCast.toUint256(int256): It Converts a signed int256 into an unsigned uint256 and checks that input must be greater than or equal to

   Status: Fixed

Low level severity issues

BCubePublicSale.sol

- [#L117] Comparison with block.timestamp: Avoid relying on block.timestamp as it can be manipulated by miners

PublicSaleTreasury.sol

- [#L32, 71, 89, 92, 95] Comparison with now: Avoid relying on now as it can be manipulated by miners

Informational

BCubePublicSale.sol

- [#L1] Using Older Solidity Compiler: Use new compiler versions to avoid bugs identified in the older versions

- [#L1] Pragma Directive mismatches with the Pragma Directives in the source files imported

Status: Fixed

- [#L144] values can be rechecked and recalculated here to make sure it's calculating the intended rates

- [#L179-185] event **LogPrivateSaleTimeExtension** in function **extendClosingTime** should be renamed to **LogPublicSaleTimeExtension** as the contract and function handles public sale

Status: Fixed

- [#L200, #L217] dollarUnits can be in decimals for specific values.

- [#L224] comment says [/// @dev stageCap is max net BCUBEs allocated until a given stage i.e. 7m, 15m for stages "pre-ico" and "ico"]: Did you mean 8m, 15m?

Status: Fixed

PublicSaleTreasury.sol

- [#L1] Using Older Solidity Compiler: Use new compiler versions to avoid bugs identified in the older versions

- [#L1] Pragma Directive mismatches with the Pragma Directives in the source files imported

Status: Fixed

- [#L24] event **LogEtherReceived(address indexed sender, uint256 value);** is already defined in BCubePublicSale contract[#L52](Event with same name and parameter types defined twice)

  Status: Fixed

- [#L36] fallback function already defined in contract BCubePublicSale

  Status: Fixed

- [#L91,94,97,100] SafeMath add can be used for calculating the allowance

  Status: Fixed

Automated Testing

Slither

Didn’t give any high severity issues

Mythril

Didn’t give any high severity issues

Smartcheck

Didn’t give any high severity issues

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines. All of the critical issues found in the initial audit have now been fixed.

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of the code. Besides a security audit, please don’t consider this report as investment advice.

b-cube.ai

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

hello@quillhash.com