



*Deliverable: Smart Contract Audit Report*

*Buyzooka Technologies*  
**Smart Contract Review**

Security Report

June 2021



## Disclaimer

The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the Company. The content, conclusions and recommendations set out in this publication are elaborated in the specific for only project.

The eNebula Solutions does not guarantee the accuracy of the data included in this study. All representations, warranties, undertakings and guarantees relating to the report are excluded, particularly concerning – but not limited to – the qualities of the assessed projects and products. Neither the Company nor any person acting on the Company's behalf may be held responsible for the use that may be made of the information contained herein.

eNebula Solutions retains the right to display audit reports and other content elements as examples of their work in their portfolio and as content features in other projects with protecting all security purpose of customer. The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities fixed - upon a decision of the Customer.

© eNebula Solutions, 2021.

# Report Summary

<b>Title</b>	Buyzooka Technologies Smart Contract Audit		
<b>Project Owner</b>	Buyzooka Technologies		
<b>Type</b>	Public		
<b>Reviewed by</b>	Vatsal Raychura	<b>Revision date</b>	12/06/2021
<b>Approved by</b>	eNebula Solutions Private Limited	<b>Approval date</b>	12/06/2021
		<b>N° Pages</b>	15

# Overview

## Background

Buyzooka Technologies requested that eNebula Solutions perform an Extensive Smart Contract audit of their Smart Contracts.

## Project Dates

The following is the project schedule for this review and report:

- **June 12:** Smart Contract Review Completed (*Completed*)
- **June 12:** Delivery of Smart Contract Audit Report (*Completed*)

## Review Team

The following eNebula Solutions team member participated in this review:

- Sejal Barad, Security Researcher and Engineer
- Vatsal Raychura, Security Researcher and Engineer

# Coverage

## Target Specification and Revision

For this audit, we performed research, investigation, and review of the smart contract of Buyzooka Technologies.

The following documentation repositories were considered in-scope for the review:

- Buyzooka Technologies Project:  
<https://github.com/Buyzooka/bzoo>

# Introduction

Given the opportunity to review Buyzooka Technologies Contracts related smart contract source code, we in the report outline our systematic approach to evaluate potential security issues in the smart contract implementation, expose possible semantic inconsistencies between smart contract code and design document, and provide additional suggestions or recommendations for improvement. Our results show that the given version of smart contracts is ready to launch after resolving the mentioned issues, there are no critical or high or medium severity issues found related to business logic, security or performance.

About Buyzooka Technologies: -

Item	Description
<b>Issuer</b>	Buyzooka Technologies
<b>Website</b>	<a href="https://www.buyzooka.io/">https://www.buyzooka.io/</a>
<b>Type</b>	ERC20
<b>Platform</b>	Solidity
<b>Audit Method</b>	Whitebox
<b>Latest Audit Report</b>	June 12, 2021

The Test Method Information:

Test method	Description
<b>Black box testing</b>	Conduct security tests from an attacker's perspective externally.
<b>Grey box testing</b>	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
<b>White box testing</b>	Based on the open-source code, non-open-source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
<b>Critical</b>	Critical severity vulnerabilities will have a significant effect on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
<b>High</b>	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
<b>Medium</b>	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
<b>Low</b>	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
<b>Weakness</b>	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

The Full List of Check Items:

Category	Check Item
<b>Basic Coding Bugs</b>	Constructor Mismatch
	Ownership Takeover
	Redundant Fallback Function
	Overflows & Underflows
	Reentrancy
	Money-Giving Bug
	Blackhole
	Unauthorized Self-Destruct
	Revert DoS
	Unchecked External Call
	Gasless Send
	Send Instead of Transfer
	Costly Loop
	(Unsafe) Use of Untrusted Libraries
	(Unsafe) Use of Predictable Variables
Transaction Ordering Dependence	
Deprecated Uses	
<b>Semantic Consistency Checks</b>	Semantic Consistency Checks
	Business Logics Review
	Functionality Checks
	Authentication Management

<b>Advanced DeFi Scrutiny</b>	Access Control & Authorization
	Oracle Security
	Digital Asset Escrow
	Kill-Switch Mechanism
	Operation Trails & Event Generation
	ERC20 Idiosyncrasies Handling
	Frontend-Contract Integration
	Deployment Consistency
	Holistic Risk Management
<b>Additional Recommendations</b>	Avoiding Use of Variadic Byte Array
	Using Fixed Compiler Version
	Making Visibility Level Explicit
	Making Type Inference Explicit
	Adhering To Function Declaration Strictly
	Following Other Best Practices

Common Weakness Enumeration (CWE) Classifications Used in This Audit:

Category	Summary
<b>Configuration</b>	Weaknesses in this category are typically introduced during the configuration of the software.
<b>Data Processing Issues</b>	Weaknesses in this category are typically found in functionality that processes data.
<b>Numeric Errors</b>	Weaknesses in this category are related to improper calculation or conversion of numbers.
<b>Security Features</b>	Weaknesses in this category are concerned with topics like authentication, access control, confidentiality, cryptography, and privilege management. (Software security is not security software.)
<b>Time and State</b>	Weaknesses in this category are related to the improper management of time and state in an environment that supports simultaneous or near-simultaneous computation by multiplesystems, processes, or threads.
<b>Error Conditions, Return Values, Status Codes</b>	Weaknesses in this category include weaknesses that occur if a function does not generate the correct return/status code, or if the application does not handle all possible return/status codes that could be generated by a function.
<b>Resource Management</b>	Weaknesses in this category are related to improper management of system resources.
<b>Behavioral Issues</b>	Weaknesses in this category are related to unexpected behaviors from code that an application uses.

<b>Business Logics</b>	Weaknesses in this category identify some of the underlying problems that commonly allow attackers to manipulate the business logic of an application. Errors in business logic can be devastating to an entire application.
<b>Initialization and Cleanup</b>	Weaknesses in this category occur in behaviors that are used for initialization and breakdown.
<b>Arguments and Parameters</b>	Weaknesses in this category are related to improper use arguments or parameters within function calls.
<b>Expression Issues</b>	Weaknesses in this category are related to incorrectly written expressions within code.
<b>Coding Practices</b>	Weaknesses in this category are related to coding practices that are deemed unsafe and increase the chances that an exploitable vulnerability will be present in the application. They may not directly introduce a vulnerability, but indicate the product has not been carefully developed or maintained.



# Findings

## Summary

Here is a summary of our findings after analyzing the Buyzooka Technologies Smart Contract Review. During the first phase of our audit, we studied the smart contract source code and ran our in-house static code analyzer through the Specific tool. The purpose here is to statically identify known coding bugs, and then manually verify (reject or confirm) issues reported by tool. We further manually review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.

Severity	No. of Issues
Critical	0
High	0
Medium	0
Low	1
Total	1

We have so far identified that there are potential issues with severity of **0 Critical, 0 High, 0 Medium, and 1 Low**. Overall, these smart contracts are well-designed and engineered, though the implementation can be improved and bug free by common recommendations given under POCs.

# Detailed Results

## Basic Code Bugs

### 1. Function Default Visibility

- SWC ID: 100
- Severity: Low
- Location: BuyzookaToken.sol
- Relationships: CWE-710: Improper Adherence to Coding Standards
- Description: The function definition of "null" lacks a visibility specifier. Note that the compiler assumes "public" visibility by default. Function visibility should always be specified explicitly to assure correctness of the code and improve readability.

```
26     constructor() ERC20("BuyzookaToken", "BZ00") {  
27         // This is the only place where we ever mint tokens.  
28         _mint(msg.sender, INITIAL_SUPPLY);  
29     }
```

- Remediations: Functions can be specified as being external, public, internal or private. It is recommended to make a conscious decision on which visibility type is appropriate for a function. This can dramatically reduce the attack surface of a contract system.

## Basic Coding Bugs

### 1. Constructor Mismatch

- Description: Whether the contract name and its constructor are not identical to each other.
- Result: Not found
- Severity: Critical

### 2. Ownership Takeover

- Description: Whether the set owner function is not protected.
- Result: Not found
- Severity: Critical

### 3. Redundant Fallback Function

- Description: Whether the contract has a redundant fallback function.
- Result: Not found
- Severity: Critical

### 4. Overflows & Underflows

- Description: Whether the contract has general overflow or underflow vulnerabilities
- Result: Not found
- Severity: Critical

### 5. Reentrancy

- Description: Reentrancy is an issue when code can call back into your contract and change state, such as withdrawing ETHs.
- Result: Not found
- Severity: Critical

### 6. Money-Giving Bug

- Description: Whether the contract returns funds to an arbitrary address.
- Result: Not found
- Severity: High

## **7. Blackhole**

- Description: Whether the contract locks ETH indefinitely: merely in without out.
- Result: Not found
- Severity: High

## **8. Unauthorized Self-Destruct**

- Description: Whether the contract can be killed by any arbitrary address.
- Result: Not found
- Severity: Medium

## **9. Revert DoS**

- Description: Whether the contract is vulnerable to DoS attack because of unexpected revert.
- Result: Not found
- Severity: Medium

## **10. Unchecked External Call**

- Description: Whether the contract has any external call without checking the return value.
- Result: Not found
- Severity: Medium

## **11. Gasless Send**

- Description: Whether the contract is vulnerable to gasless send.
- Result: Not found
- Severity: Medium

## **12. Send Instead of Transfer**

- Description: Whether the contract uses send instead of transfer.
- Result: Not found
- Severity: Medium

## **13. Costly Loop**

- Description: Whether the contract has any costly loop which may lead to Out-Of-Gas exception.

- Result: Not found
- Severity: Medium

#### **14. (Unsafe) Use of Untrusted Libraries**

- Description: Whether the contract use any suspicious libraries.
- Result: Not found
- Severity: Medium

#### **15. (Unsafe) Use of Predictable Variables**

- Description: Whether the contract contains any randomness variable, but its value can be predicated.
- Result: Not found
- Severity: Medium

#### **16. Transaction Ordering Dependence**

- Description: Whether the final state of the contract depends on the order of the transactions.
- Result: Not found
- Severity: Medium

#### **17. Deprecated Uses**

- Description: Whether the contract use the deprecated tx.origin to perform the authorization.
- Result: Not found
- Severity: Medium

#### **Semantic Consistency Checks**

- Description: Whether the semantic of the white paper is different from the implementation of the contract.
- Result: Not found
- Severity: Critical

## Conclusion

In this audit, we thoroughly analyzed Buyzooka Technologies Smart Contracts. The current code base is well organized but there are promptly some low-level issues found in this phase of Smart Contract Audit.

Meanwhile, we need to emphasize that smart contracts as a whole are still in an early, but exciting stage of development. To improve this report, we greatly appreciate any constructive feedbacks or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

## About eNebula Solutions

We believe that people have a fundamental need to security and that the use of secure solutions enables every person to more freely use the Internet and every other connected technology. We aim to provide security consulting service to help others make their solutions more resistant to unauthorized access to data & inadvertent manipulation of the system. We support teams from the design phase through the production to launch and surely after.

The eNebula Solutions team has skills for reviewing code in C, C++, Python, Haskell, Rust, Node.js, Solidity, Go, and JavaScript for common security vulnerabilities & specific attack vectors. The team has reviewed implementations of cryptographic protocols and distributed system architecture, including in cryptocurrency, blockchains, payments, and smart contracts. Additionally, the team can utilize various tools to scan code & networks and build custom tools as necessary.

Although we are a small team, we surely believe that we can have a momentous impact on the world by being translucent and open about the work we do.

For more information about our security consulting, please mail us at – [contact@enebula.in](mailto:contact@enebula.in)