ALPHA APE LLC

GrowBananas BNB Pool

Audit Report
May.12.2022

Table of Contents

SUMMARY OF AUDIT

DETAILS OF AUDITED PROJECT

Audited Project:	Grow Bananas
Source Code:	https://www.bscscan.com/address/0x3F3c792448dd1ad55A21a5627628dB007795E927#code
Solidity File:	GrowBananas.sol
Security Audit Date:	May. 12 - 2022
Revisions:	None
Auditing Methods:	Automatic review + Manual review

AUDITING METHODS AND COVERING SECTORS

Evaluation objective for the security audit:

- Quality of smart contracts code
- Issues and vulnerabilities with security
- Documentation, project specifics and commenting on smart contract
- Correctness of specifications regarding the use-case
- Following the best practices on smart contract

Audit covers these sectors of smart contract for possible vulnerabilities, issues and recommendations for better practices in case of severe or medium issues:

- Dependence Transaction Order
- Single and Cross-Function Reentrancy
- Time Dependency
- Integer Overflow
- Integer Underflow
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Number rounding errors
- Malicious contract behaviour and abusable functions
- Possible DoS vulnerabilities
- Insufficient gas issues
- Logical oversights
- Access control
- Centralization of power
- Logic-Specification
- Contradiction
- Functionality duplication

The code review conducted for this audit follows the following structure:

1. Review of the specifications, documentation and commenting provided by the project owners regarding the functionality of the smart contract
2. Automated analysis of the smart contract followed by manual, line-by-line analysis of the smart contract
3. Assessment of smart contracts correctness regarding the documentation and commenting compared to functionality
4. Assessment of following the best practices
5. Recommendations for better practises in case severe or medium vulnerabilities

SMART CONTRACT DETAILS

Contract ID	0x3F3c792448dd1ad55A21a5627628dB007795E927
Blockchain:	Binance Smart Chain
Language Used:	Solidity
Compiler Version:	v0.8.9+commit.e5eed63a
Bscscan Verification:	2022-05-11
Type of Smart Contract:	Pooling / Utilities
Libraries Used:	Unverified
Optimization Enabled:	Yes with 200 runs

SUMMARY OF AUDIT RESULTS

Alpha Ape Network smart contract audit for Grow Bananas is marked as **PASSED** result without severe issues on the logic and functions of the contract. Review of the documentation and description of the projects use-case and the smart contract follows the line of good practices.

SEVERITY OF RISKS AND VULNERABILITIES

LOW-SEVERITY	MEDIUM-SEVERITY	HIGH-SEVERITY
1	**1**	**0**

REPORTED VULNERABILITIES AND ISSUES

LEVEL OF SEVERITY	DESCRIPTION	FILE	CODELINES AFFECTED
LOW-SEVERITY	A control flow decision is made based on the block.timestamp environment variable.	GrowBananas.sol	L: 282 C: 15
MEDIUM-SEVERITY	Multiple calls are executed in the same transaction.	GrowBananas.sol	L: 219 C: 8

FINDINGS IN-DEPTH

LOW-SEVERITY CONTROL FLOW DECISION IS MADE BASED ON THE BLOCK.TIMESTAMP ENVIRONMENT VARIABLE

Analysis: The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner.

Possible vulnerability: Malicious miners can alter the timestamp of their blocks, especially if they can gain advantages by doing so. However, miners can't set a timestamp smaller than the previous one, nor can they set the timestamp too far ahead in the future.

Recommendation: Team is recommended to write the smart contract with the notion that block values are not precise, and the use of them can lead to unexpected effects. Alternatively, team may use oracles to overcome this possible vulnerability.

GrowBananas.sol L: 282 C: 15

MEDIUM-SEVERITY MULTIPLE CALLS ARE EXECUTED IN THE SAME TRANSACTION

Analysis: Call is executed following another call within the same transaction. It is possible that the call never gets executed if a prior call fails permanently. This might be caused intentionally by a malicious callee. Refactoring the code to such that each transaction only executes one external call or making sure that all callees can be trusted overcomes this vulnerability.

Possible vulnerability: External calls can fail accidentally or deliberately, which can cause a DoS condition in the contract. To minimize the damage caused by such failures, it is better to isolate each external call into its own transaction that can be initiated by the recipient of the call.

Recommendation: It is recommended to follow call best practices:

- Avoid combining multiple calls in a single transaction, especially when calls are executed as part of a loop
- Implement the contract logic to handle failed calls

GrowBananas.sol L: 219 C: 8

DOCUMENTATION AND COMMENTING

The code has a minimal amount of comments and documentation. Improving stage of commenting on smart contracts helps userbase to understand all the functionalities of the contract. Functions are marked in understandable way.

CORRECTNESS OF SPECIFICATIONS

Smart contract follows the functionality that is stated in the documentation and description of the contract. The use-case is also in line what is described about the project.

FOLLOWING THE BEST PRACTICES

The smart contract follows the best practices in majority and major parts follow the standards of BEP20 contract. Minor parts that do not follow the best practices, or is affected by possible vulnerabilities, do not rise fear in malicious use of the contract or severe issues in projects use-case.

HISTORY OF REVISIONS, FUNCTIONS AND VARIABLES

HISTORY OF REVISIONS

Initial audit was performed May. 12-2022. No further revision history.

Team has been provided recommendations for better practices for revision. Conclusion with security audit performer, the team has decided to leave contract as is. Respecting the teams decision, there is no major issues were found to cause issues or malicious use of the contract and remains as-is.

PUBLIC AND PRIVATE FUNCTIONS AND VARIABLES

Function	Parameters	Visibility	Modifiers	Returns	Requires	Events
constructor						
regrowBananas	address ref	public			initialized	
sellBananas		public			initialized	
bananaRewards	address ref	public		uint256		
buyBananas	address ref	public	payable		initialized	
calculateTrade	uint256 rt uint256 rs uint256 bs	private		uint256		
calculateBananaSell	uint256 eggs	public		uint256		
calculateBananaBuy	uint256 eth,uint256 contractBalance	public		uint256		
calculateBananaBuySimple	uint256 eth	public		uint256		
devFee	uint256 amount	private		uint256		
LetsGrow		public	payable onlyOwner		marketBananas s == 0	
getBalance		public		uint256		
getMyGrownBananas	address adr	public		uint256		
getMyBananas	address adr	public		uint256		
getBananasSinceLastHatch	address adr	public		uint256		
min	uint256 a, uint256 b	private		uint256		

Variable	Type	Visibilty	Read by Functions	Written by Functions
BANANAS_TO_GROW_TO_PICK	**uint256**	private		
PSN	**uint256**	private		
PSNH	**uint256**	private		
devFeeVal	**uint256**	private		
initialized	**bool**	private		
recAdd	**address payable**	private		
growingBananas	mapping (address => uint256)	private		
pickedBananas	mapping (address => uint256)	private		
lastPick	mapping (address => uint256)	private		
referrals	mapping (address => address)	private		
marketBananas	uint256	private		