Alpha Ape LLC

Security Audit report for
DecentraBNB

Ethereum Contract Address
0x833850be8858722cfc5e5e75f2fe6275e055d888

August.03.2022

Table of Contents

SUMMARY OF AUDIT

DETAILS OF AUDITED PROJECT

Audited Project:	DecentraBNB – (DBNB)
Source Code:	https://etherscan.io/token/0x833850be8858722cfc5e5e75f2fe6275e055d888#code (verified)
Solidity File:	decentrabnb.sol
Security Audit Date:	Aug. 3 - 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
- Insufficient gas issues
- Logical oversights
- Access control
- Centralization of power
- Logic-Specification
- Contradiction
- Functionality duplication
- Malicious contract behaviour and abusable functions
- Possible DoS vulnerabilities

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	0x833850be8858722cfc5e5e75f2fe6275e055d888 (verified)
Blockchain:	Ethereum
Language Used:	Solidity
Compiler Version:	v0.8.13+commit.abaa5c0e
Bscscan Verification:	2022-06-30
Type of Smart Contract:	ERC20 Token - Utility
Libraries Used:	
Optimization Enabled:	Yes with 200 runs

SUMMARY OF AUDIT RESULTS

Alpha Ape LLC smart contract audit for DecentraBNB (DBNB) 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 contract follows the line of good practices for majority. Clean and default commented contract.

SEVERITY OF RISKS AND VULNERABILITIES

LOW-SEVERITY	MEDIUM-SEVERITY	HIGH-SEVERITY
3	0	0

REPORTED VULNERABILITIES AND ISSUES

LEVEL OF SEVERITY	DESCRIPTION	FILE	CODELINES AFFECTED
LOW-SEVERITY	A floating pragma is set.	decentrabnb.sol	L: 15 C: 0
LOW-SEVERITY	Potential use of "block.number" as source of randonmness.	decentrabnb.sol	L: 305 C: 16
LOW-SEVERITY	Potential use of "block.number" as source of randonmness.	decentrabnb.sol	L: 539 C: 23

FINDINGS IN-DEPTH

LOW-SEVERITY A FLOATING PRAGMA IS SET

```
13 */
14
15 pragma solidity ^0.8.13;
16
17 abstract contract Context {
```

Analysis: The current pragma Solidity directive is ""^0.8.13"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Possible vulnerability: Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Recommendation: Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen. Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile locally.

decentrabnb.sol L: 15 C: 0

LOW-SEVERITY Potential use of "block.number" as source of randonmness.

```
303 require(!_bots[from] && !_bots[to]);
304
305 if (block.number <= _launchBlock + 4) {
306 if (from != uniswapV2Pair && from != address(uniswapV2Router)) {
307 _bots[from] = true;
```

Analysis: The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Possible vulnerability: Use of block.timestamp or block.number is by default insecure, as a malicious miner can choose to provide any timestamp within a few seconds, and still get his block accepted by others. Use of blockhash, block.difficulty and other fields is also insecure, as they're controlled by the miner.

Recommendation: No actual vulnerability in the audited contract, leaving the low-severity finding as a note of not following the actual best practices. No adjustments recommended.

decentrabnb.sol L: 305 C: 16

LOW-SEVERITY Potential use of "block.number" as source of randonmness.

```
537 require(!_tradingOpen,"trading is already open");
538 _tradingOpen = true;
539 _launchBlock = block.number;
540 }
```

Analysis: The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Possible vulnerability: Use of block.timestamp or block.number is by default insecure, as a malicious miner can choose to provide any timestamp within a few seconds, and still get his block accepted by others. Use of blockhash, block.difficulty and other fields is also insecure, as they're controlled by the miner.

Recommendation: No actual vulnerability in the audited contract, leaving the low-severity finding as a note of not following the actual best practices. No adjustments recommended.

decentrabnb.sol L: 539 C: 23

DOCUMENTATION AND COMMENTING

The code has a decent amount of comments and documentation. Improving stage of commenting on smart contracts, and cleaning the commenting, helps userbase to understand all the functionalities and use-case of the contract. Functions are marked in understandable way and commenting on the contract has been made very human readable to understand.

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. Verified as an real-life utility as going through the projects website and interviewing the core-team.

FOLLOWING THE BEST PRACTICES

The contract follows the best practices in majority. 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. Low-level severities that are noted in the audit are only improper adherence to coding standards, than security vulnerabilities.

HISTORY OF REVISIONS, FUNCTIONS AND VARIABLES

HISTORY OF REVISIONS

Initial audit was performed Aug. 3-2022. No further revision history.

Team has been provided recommendations for better practices for revision. There is no severe or medium notifications or recommendations for the revision. Contract is deployed and immutable and the possible vulnerabilities has been stated in the security audit. Project is following its use case and security auditor has validated the use-case of the contract to follow its description the best possible way.