VITALBlock security.

Blockchain Security | Smart Contract Audit | KYC Certification | SAFU | CEX Listing | Marketing

MADE IN CANADA

BNB GOAT

AUDIT

SECURITY ASSESSMENT

25th September 2025

For

Making Blockchain, Defi And Web3 A Safer Place.

Smart Check | SLITHER | TRAIL OF BITS | MythX

@VB_Audit | @Vitalblock | @VB_Audit | www.vitalblock.org

CONTENTS

INTRODUCTION

Auditing Firm	**VITAL BLOCK SECURITY**
Client Firm	BNB GOAT
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Contract Address	0x88883075fb050f4199cF7b2011cC15f1c966403f
Source Code Light	Verified
Centralization	Active ownership
Compiler Version	v0.8.23+commit.f704f362
Blockchain	**BINANCE CHAIN**
Website	https://thebnbgoat.com/
Twitter	https://x.com/THEBNBGOAT
Telegram	https://t.me/TheBNBGOAT
Medium	https://thebnbgoat.com/assets/$BGOAT%20Whitepaper%20.pdf
Prelim Report Date	September 25TH 2025
Final Report Date	September 25TH 2025

ℹ Verify the authenticity of this report on our GitHub Repo: https://www.github.com/vital-block

Document Properties

Client	BNBGOAT
Title	Smart Contract Audit Report
Target	BNBGOAT
Version	1.0
Author	Akhmetshin Marat
Auditors	Akhmetshin Marat, James BK, Ben Partrick , C. John
Reviewed by	Dima Meru
Approved by	Prince Mitchell
Classification	Public

Version Info

Version	Date	Author(s)	Description
1.0	September 25TH , 2025	C. John	Final Release
1.0-AP	September 25TH , 2025	C. John	Release Candidate

Contact

For more information about this document and its contents, please contact Vital Block Security Inc.

Name	Akhmetshin Marat
Phone ☎	+1 (579) 817-7049
Email	info@vitalblock.org

In the following, we show the specific pull request and the commit hash value used in this audit.

- BNB GOAT (BFFG60877)
- https://bscscan.com/token/0x88883075fb050f4199cf7b2011cc15f1c966403f (GBHHO764)

About Vital Block Security

Vital Block Security provides professional, thorough, fast, and easy-to-understand smart contract security audit. We do in-depth and penetrative static, manual, automated, and intelligent analysis of the smart contract. Some of our automated scans include tools like ConsenSys MythX, Mythril, Slither, Surya. We can audit custom smart contracts, DApps, NFTs, etc (including the service of smart contract auditing). We are reachable at Telegram (https://t.me/vitalblock), Twitter (http://twitter.com/Vb_Audit), or Email (info@vitalblock.org).

Table 1.2: Vulnerability Severity Classification

Impact \ Likelihood	High	Medium	Low
High	Critical	High	Medium
Medium	High	Medium	Low
Low	Medium	Low	Low

Methodology

To standardize the evaluation, we define the following terminology based on the OWASP Risk Rating Methodology.

- Likelihood represents how likely a particular vulnerability is to be uncovered and exploited in the wild;
- Impact measures the technical loss and business damage of a successful attack;
- Severity demonstrates the overall criticality of the risk.

SCOPE OF WORK

Vital Block was consulted by BNBGOAT to conduct the smart contract audit of its. SOLIDITY (SOL) source code. The audit scope of work is strictly limited to the mentioned .Sol file only:

O.BGOAT.SOL

ℹ **External contracts and/or interfaces dependencies are not checked due to being out of scope.**

Verify audited contract's contract address and deployed link below:

Public Contract Address	
0x88883075fb050f4199cF7b2011cC15f1c966403f	
Contract Name	BNB GOAT
Ticker	**$BGOAT**
Total Supply	**1,000,000,000**

AUDIT METHODOLOGY

Smart contract audits are conducted using a set of standards and procedures. Mutual collaboration is essential to performing an effective smart contract audit. Here's a brief overview of Vital Block Security auditing process and methodology:

CONNECT

- **The onboarding team gathers source codes, and specifications to make sure we understand the size, and scope of the smart contract audit.**

AUDIT

- **Automated analysis is performed to identify common contract vulnerabilities. We may use the following third-party frameworks and dependencies to perform the automated analysis:**
  - **Remix IDE Developer Tool**
  - **Open Zeppelin Code Analyzer**
  - **SWC Vulnerabilities Registry**
  - **DEX Dependencies, e.g., Pancakeswap, Uniswap**
- **Simulations are performed to identify centralized exploits causing contract and/or trade locks.**
- **A manual line-by-line analysis is performed to identify contract issues and centralized privileges. We may inspect below mentioned common contract vulnerabilities, and centralized exploits:**

Centralized Exploits	○ **Token Supply Manipulation** ○ **Access Control and Authorization** ○ **Assets Manipulation** ○ **Ownership Control** ○ **Liquidity Access** ○ **Stop and Pause Trading** ○ **Ownable Library Verification**

Common Contract Vulnerabilities

- **Integer Overflow**
- **Lack of Arbitrary limits**
- **Incorrect Inheritance Order**
- **Typographical Errors**
- **Requirement Violation**
- **Gas Optimization**
- **Coding Style Violations**
- **Re-entrancy**
- **Third-Party Dependencies**
- **Potential Sandwich Attacks**
- **Irrelevant Codes**
- **Divide before multiply**
- **Conformance to Solidity Naming Guides**
- **Compiler Specific Warnings**
- **Language Specific Warnings**

REPORT

- **The auditing team provides a preliminary report specifying all the checks which have been performed and the findings thereof.**
- **The client's development team reviews the report and makes amendments to the codes.**
- **The auditing team provides the final comprehensive report with open and unresolved issues.**

PUBLISH

- **The client may use the audit report internally or disclose it publicly.**

ℹ **It is important to note that there is no pass or fail in the audit, it is recommended to view the audit as an unbiased assessment of the safety of solidity codes.**

Table 1.0 The Full Audit Checklist

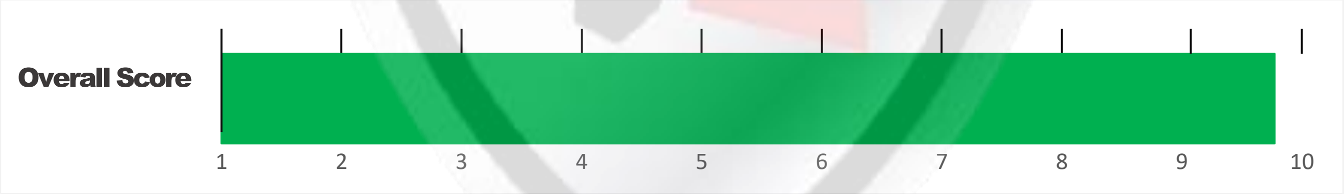
Category	Checklist Items
Basic Coding Bugs	Constructor Mismatch
	Ownership Takeover
	Redundant Fallback Function
	Overflows & Underflows
	Reentrancy
	Money-Giving Bug
	Blackhole
	Unauthorized Self-Destruct
	Revert DoS
	Unchecked External Call
	Gasless Send
	Send Instead Of Transfer
	Costly Loop
	(Unsafe) Use Of Untrusted Libraries
	(Unsafe) Use Of Predictable Variables
	Transaction Ordering Dependence
	Deprecated Uses
Semantic Consistency Checks	Semantic Consistency Checks
Advanced DeFi Scrutiny	Business Logics Review
	Functionality Checks
	Authentication Management
	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
Additional Recommendations	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

EXECUTIVE SUMMARY

Vital Block Security has performed the automated and manual analysis of the BNB GOAT Sol code. The code was reviewed for common contract vulnerabilities and centralized exploits.
Here's a quick audit summary:

Status	Critical ! 🔴	Major " 🟠	Medium # 🟡	Minor $ 🟢	Unknown % 🟤
Open	0	0	1	2	0
Acknowledged	0	0	0	1	0
Resolved	0	0	0	0	0
Noteworty OnlyOwner Privileges	Set Taxes and Ratios, Airdrop, Set Protection Settings, Set Reward Properties, Set Reflector Settings, Set Swap Settings, Set Pair and Router				

BNB GOAT Smart contract has achieved the following score: 98.0



i Please note that smart contracts deployed on blockchains aren't resistant to exploits, vulnerabilities and/or hacks. Blockchain and cryptography assets utilize new and emerging technologies. These technologies present a high level of ongoing risks. For a detailed understanding of risk severity, source code vulnerability, and audit limitations, kindly review the audit report thoroughly.

i Please note that centralization privileges regardless of their inherited risk status - constitute an elevated impact on smart contract safety and security.

RISK CATEGORIES

Smart contracts are generally designed to hold, approve, and transfer tokens. This makes them very tempting attack targets. A successful external attack may allow the external attacker to directly exploit. A successful centralization-related exploit may allow the privileged role to directly exploit. All risks which are identified in the audit report are categorized here for the reader to review:

Risk Type	Definition
Critical !🔴	These risks could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
Major "🟠	These risks are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to high-risk severity.
Medium #🟡	These risks should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution. Low-risk re-entrancy-related vulnerabilities should be fixed to deter exploits.
Minor $🟢	These risks do not pose a considerable risk to the contract or those who interact with it. They are code-style violations and deviations from standard practices. They should be highlighted and fixed nonetheless.
Unknown %🟤	These risks pose uncertain severity to the contract or those who interact with it. They should be fixed immediately to mitigate the risk uncertainty.

All statuses which are identified in the audit report are categorized here for the reader to review:

Status Type	Definition
Open	Risks are open.
Acknowledged	Risks are acknowledged, but not fixed.
Resolved	Risks are acknowledged and fixed.

CENTRALIZED PRIVILEGES

Centralization risk is the most common cause of cryptography asset loss. When a smart contract has a privileged role, the risk related to centralization iselevated.

There are some well-intended reasons haveprivileged roles, such as:

- **Privileged roles can be granted the power to** pause()**the contract in case of an external attack.**
- **Privileged roles can use functions like,** include(), **and** exclude() **to add or remove wallets from fees, swap checks, and transaction limits. This is useful to run a presale and to list on an exchange.**

Authorizing privileged roles to externally-owned-account (EOA) is dangerous. Lately, centralization-related losses are increasing in frequency and magnitude.

- **The client can lower centralization-related risks by implementing below mentioned practices:**
- **Privileged role's private key must be carefully secured to avoid any potential hack.**
- **Privileged role should be shared by multi-signature (multi-sig) wallets.**
- **Authorized privilege can be locked in a contract, user voting, or community DAO can be introduced to unlock the privilege.**
- **Renouncing the contract ownership, and privileged roles.**
- **Remove functions with elevated centralization risk.**

ℹ **Understand the project's initial asset distribution. Assets in the liquidity pair should be locked. Assets outside the liquidity pair should be locked with a release schedule.**

AUTOMATED ANALYSIS

Symbol	Definition
🛑	**Function modifies state**
💵	**Function is payable**
🔒	**Function is internal**
🔐	**Function is private**
❗	**Function is important**

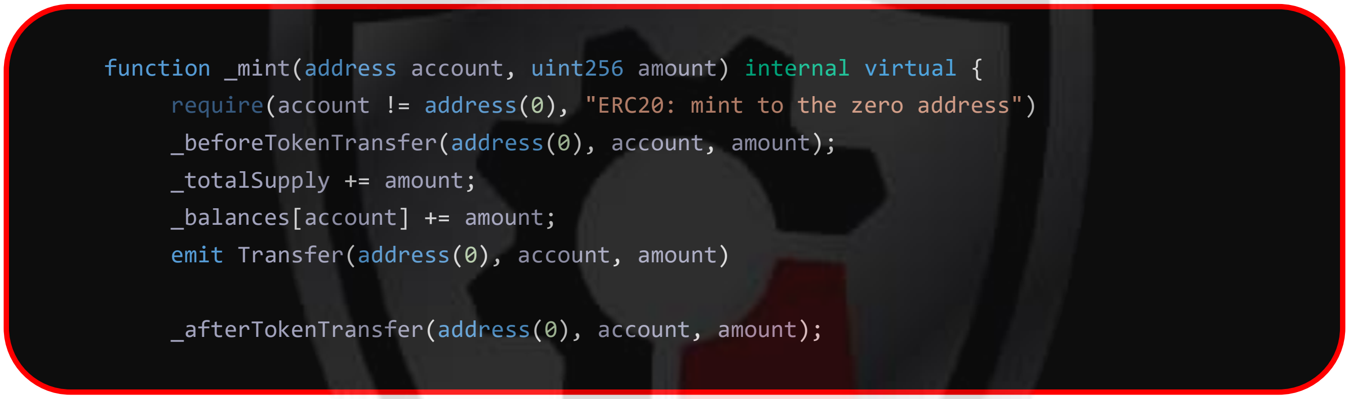
```
| **BNB GOAT** | Interface |  |||
| └ | totalSupply | External ❗ |  |NO❗ |
| └ | decimals | External ❗ |  |NO❗ |
| └ | symbol | External ❗ |  |NO❗ |
| └ | name | External ❗ |  |NO❗ |
| └ | getOwner | External ❗ |  |NO❗ |
| └ | balanceOf | External ❗ |  ❗  |NO❗ ❗
| └ | transfer | External ❗ | "  ❗  🛑 |NO❗ ❗
| └ | allowance | External ❗ |  ❗  |NO❗ ❗
| └ | approve | External ❗ | "  ❗  🛑 |NO❗ ❗
| └ | transferFrom | External ❗ | "  |NO❗ |
||||||
| **IFactoryV2** | Interface |  |||
| └ | getPair | External ❗ |  |NO❗ |
| └ | createPair | External ❗ | "  |NO❗ |
||||||
| **IV2Pair** | Interface |  |||
| └ | factory | External ❗ |  |NO❗ |
| └ | getReserves | External ❗ |  |NO❗ |
| └ | sync | External ❗ | "  |NO❗ |
||||||
| **IRouter01** | Interface |  |||
| └ | factory | External ❗ |  |NO❗ |
| └ | BNB| External ❗ |  |NO❗ |
| └ | addLiquidityBNB| External ❗ |  # |NO❗ |
| └ | addLiquidity | External ❗ | " |NO❗ |
| └ | swapExacBNBTokens | External ❗ |  # |NO❗ |
| └ | getAmountsOut | External ❗ |  |NO❗ |
| └ | getAmountsIn | External ❗ |  |NO❗ |
||||||
| **IRouter02** | Interface | IRouter01 |||
| └ | swapExactTokensForBNBSupportingFeeOnTransferTokens | External ❗ | " |NO❗ |
| └ | swapExactBNBForTokensSupportingFeeOnTransferTokens | External ❗ |  # |NO❗ |
| └ | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ❗ | " ! 🛑 |NO❗ |
| └ | swapExactTokensForTokens | External ❗ | " |NO❗ |
||||||
| **Protections** | Interface |  |||
| └ | checkUser | External ❗ | " ! 🛑 |NO❗ |
| └ | setLaunch | External ❗ | " |NO❗ |
| └ | setLpPair | External ❗ | " |NO❗ |
| └| BGOAT | External ❗ | "! 🛑 |NO❗ !
| └ | removeSniper | External ❗ |!" 🛑 |NO❗ |
||||||
| **Cashier** | Interface |  |||
| └ | setRewardsProperties | External ❗ | " |NO❗ |
| └ | tally | External ❗ ! " 🛑 |NO❗ |
| └ | load | External ❗ ! 💵 |NO❗ |
| └ | cashout | External ❗ | " ! 🛑 |NO❗ |
| └ | giveMeWelfarePlease | External ❗ | " ! 🛑 |NO❗ |
| └ | getTotalDistributed | External ❗ | ! |NO❗ |
| └ | getUserInfo | External ❗ | ! |NO❗ |
| └ | getUserRealizedRewards | External ❗ | ! |NO❗ |
| └ | getPendingRewards | External ❗ | ❗ |NO❗ |
| └ | initialize | External ❗ | " ❗ 🛑 |NO❗ |
| └ | getCurrentReward | External ❗ | |NO❗ |
||||||
| **BNB ** | Implementation | SafeMath |||
| └ | <Constructor> | Public ❗ | ❗ #💵 |NO❗❗|
| └ | transferOwner | External ❗ | " ❗ 🛑 | onlyOwner |
| └ | renounceOwnership | External ❗ | " ❗ 🛑 | NO ❗
| └ | setOperator | Public ❗ | " ❗ 🛑 |NO❗ |
| └ | renounceOriginalDeployer | External ❗ | " 🛑 |NO❗ |
| └ | <Receive WBNB> | External ❗ | ❗ #💵 |NO❗❗|
| └ | totalSupply | External ❗ | ❗ |NO❗ |
| └ | decimals | External ❗ | ❗ |NO❗ |
| └ | symbol | External ❗ | ❗ |NO❗ |
| └ | name | External ❗ | ❗ |NO❗ |
| └ | getOwner | External ❗ | ❗ |NO❗ |
| └ | balanceOf | Public ❗ | ❗ |NO❗ |
| └ | allowance | External ❗ | ❗ |NO❗ |
| └ | approve | External ❗ | " ❗ 🛑 |NO❗ |
| └ | _approve | Internal $ | "🔒 🛑 | |
| └ | approveContractContingency | Public ❗ | " ❗ 🛑 | onlyOwner |
| └ | transfer | External ❗ | " ❗ 🛑 |NO❗ |
| └ | transferFrom | External ❗ | " ❗ 🛑 |NO❗ |
| └ | setNewRouter | External ❗ | " ❗ 🛑 | onlyOwner |
| └ | setLpPair | External ❗ | " ❗ 🛑 | onlyOwner |
| └ | setInitializers | External ❗ | " ❗ 🛑 | onlyOwner |
| └ | isExcludedFromFees | External ❗ | ❗ |NO❗ |
| └ | isExcludedFromDividends | External ❗ | ❗ |NO❗ |
| └ | isExcludedFromProtection | External ❗ | ❗ |NO❗ |
| └ | setDividendExcluded | Public ❗ | " ❗ 🛑 | onlyOwner |
| └ | setExcludedFromFees | Public ❗ | " ❗ 🛑 | onlyOwner |
```

BNB GOAT - 01 POSSIBLE OVERFLOW

Category	Severity ●	Location	Status
Status Mathematical Operations	**Minor**	./src/BGOAT.Sol	**Acknowledged**

Description

In **updateForMinter ,** the following equation is used inside an unchecked block



```
function _mint(address account, uint256 amount) internal virtual {
     require(account != address(0), "ERC20: mint to the zero address")
     _beforeTokenTransfer(address(0), account, amount);
     _totalSupply += amount;
     _balances[account] += amount;
     emit Transfer(address(0), account, amount)

     _afterTokenTransfer(address(0), account, amount);
```

Minter can **Not** issue more $BGOAT tokens indefinitely.
Note that as of the date of publishing, the above review reflects the current understanding of known security patterns as they relate to the $BGOAT contract.

Recommendation

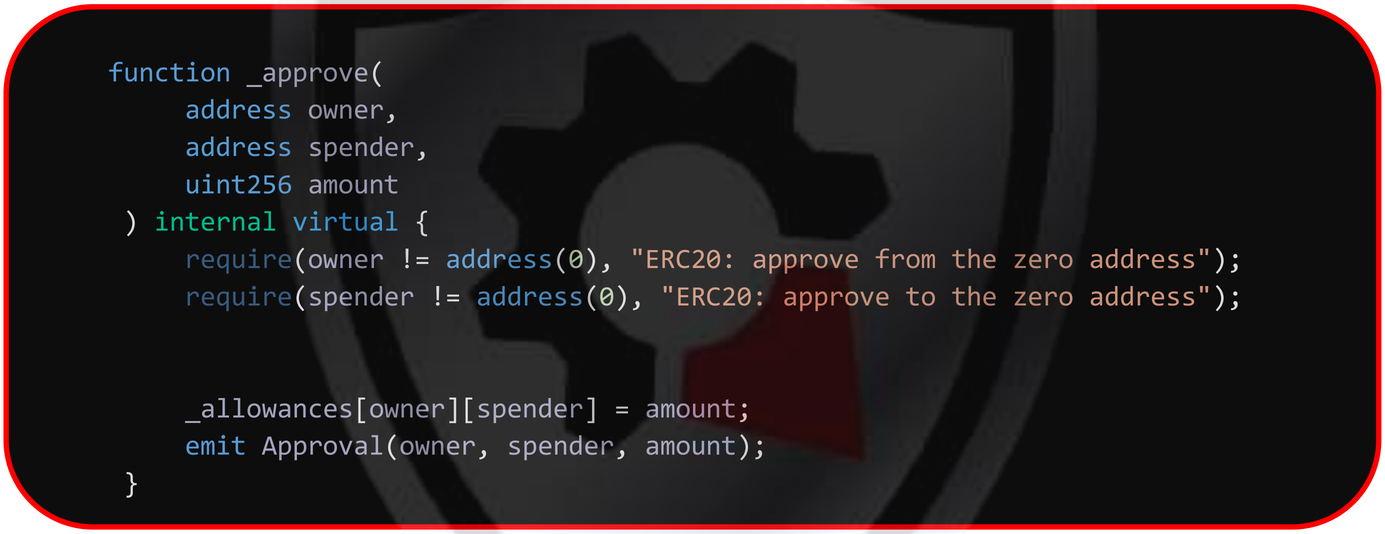
We recommend either checking for overflow in this case, or ensuring that the **PairsIn** is close enough it will never cause an overflow.

BNB GOAT - 02 POSSIBLE OVERFLOW

Category	Severity ●	Location	Status
Inconsistency	**Informational**	./src/BGOAT.Sol	Acknowledged

Description

In **updateForOwner**, Relevant Function Snippet



```
function _approve(
     address owner,
     address spender,
     uint256 amount
 ) internal virtual {
     require(owner != address(0), "ERC20: approve from the zero address");
     require(spender != address(0), "ERC20: approve to the zero address");


     _allowances[owner][spender] = amount;
     emit Approval(owner, spender, amount);
 }
```

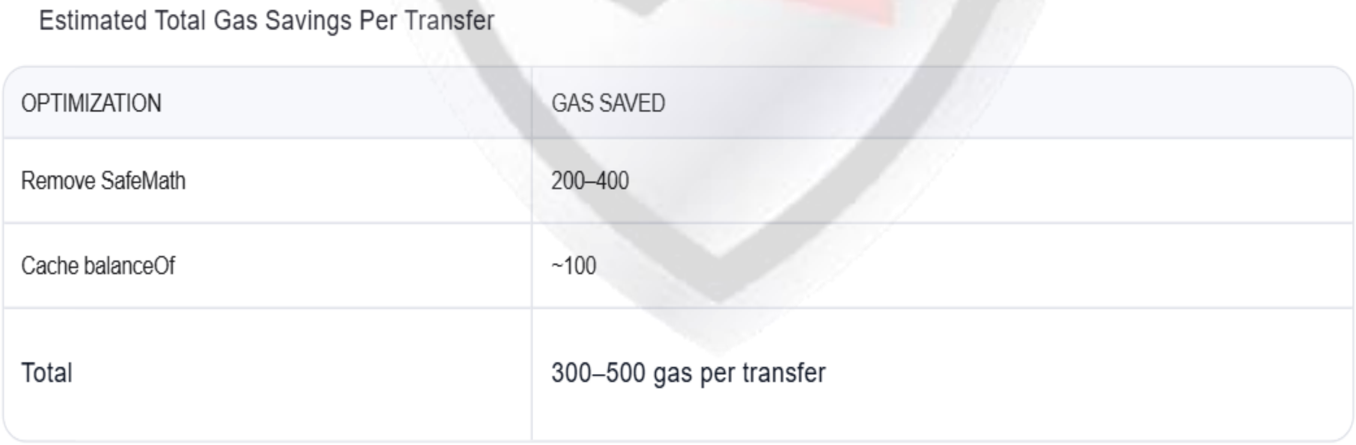
To ensure ownership efficiency, the BNB GOAT Team has implemented a reserve cache mechanism. This system standardizes the procedures for managing reserve ownership data across various scenarios, including tax generation, data updates, and final persistence.

Recommendation

Revise the above functions to following a consistent approach to use the reserve cache mechanism.

OPTIMIZATIONS | $BGOAT

ID	Title	Category	Status
ERR	**Logarithm Refinement Optimization**	Gas Optimization	Acknowledged ●
YUU	**Checks Can Be Performed Earlier**	Gas Optimization	Acknowledged ●
BGH	**Unnecessary Use Of SafeMath**	Gas Optimization	Acknowledged ●
JUP	**Struct Optimization**	Gas Optimization	Acknowledged ●
WEE	**Unused State Variable**	Gas Optimization	Acknowledged ●



Estimated Total Gas Savings Per Transfer

OPTIMIZATION	GAS SAVED
Remove SafeMath	200–400
Cache balanceOf	~100
Total	300–500 gas per transfer

General Detectors

Missing Zero Address Validation

Some functions in this contract may not appropriately check for zero addresses being used.

⚠ Attention Required

Consistent Solidity Version

This contract uses a conventional or very New version of Sol dependency

⚠ Attention Required

- ✓ No compiler version inconsistencies found
- ✓ No unchecked call responses found
- ✓ No vulnerable self-destruct functions found
- ✓ No assertion vulnerabilities found
- ✓ No old solidity code found
- ✓ No external delegated calls found
- ✓ No external call dependency found
- ✓ No vulnerable authentication calls found
- ✓ No invalid character typos found
- ✓ No RTL characters found
- ✓ No dead code found
- ✓ No risky data allocation found
- ✓ No uninitialized state variables found
- ✓ No uninitialized storage variables found
- ✓ No vulnerable initialization functions found
- ✓ No risky data handling found
- ✓ No number accuracy bug found
- ✓ No out-of-range number vulnerability found
- ✓ No map data deletion vulnerabilities found
- ✓ No tautologies or contradictions found
- ✓ No faulty true/false values found
- ✓ No innacurate divisions found
- ✓ No redundant constructor calls found
- ✓ No vulnerable transfers found
- ✓ No vulnerable return values found
- ✓ No uninitialized local variables found
- ✓ No default function responses found
- ✓ No missing arithmetic events found
- ✓ No missing access control events found
- ✓ No redundant true/false comparisons found
- ✓ No state variables vulnerable through function calls found
- ✓ No buggy low-level calls found
- ✓ No expensive loops found
- ✓ No bad numeric notation practices found
- ✓ No missing constant declarations found
- ✓ No missing external function declarations found
- ✓ No vulnerable payable functions found
- ✓ No vulnerable message values found

Vulnerability Scan

REENTRANCY

✓ No reentrancy risk found

Severity	Minor
Confidence Parameter	Certain

Vulnerability Description

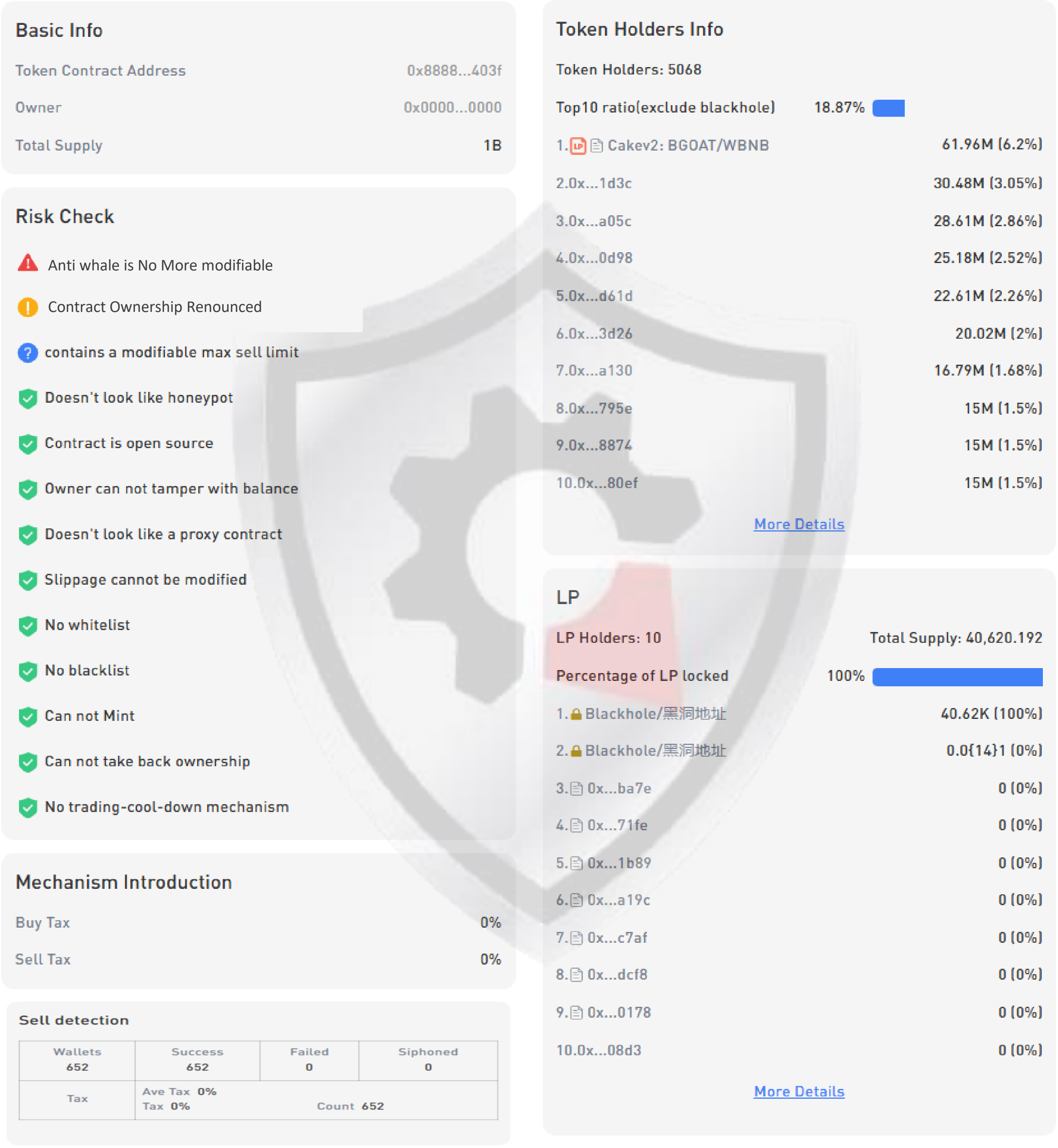
✓ **RENOUNCED:** No additional amount of BGOAT token can be minted by a private wallet or contract.
(Which is normal for major contract utility options)

Scanning Line:

```
    function renounceOwnership   public virtual onlyOwner {
        _transferOwnership address(0)
    }

    /**
     * @dev Transfers ownership of the contract to a new account
(`newOwner`).
     * Can only be called by the current owner.
     */
    function transferOwnership(address newOwner) public virtual
onlyOwner {
        require(
            newOwner    address(0),
            "Ownable: new owner is the zero address"
        );
        _transferOwnership(newOwner)
    }
```

Auto Contract Scan



Basic Info

Token Contract Address	0x8888...403f
Owner	0x0000...0000
Total Supply	1B

Risk Check

- ⚠ Anti whale is No More modifiable
- ! Contract Ownership Renounced
- ? contains a modifiable max sell limit
- ✓ Doesn't look like honeypot
- ✓ Contract is open source
- ✓ Owner can not tamper with balance
- ✓ Doesn't look like a proxy contract
- ✓ Slippage cannot be modified
- ✓ No whitelist
- ✓ No blacklist
- ✓ Can not Mint
- ✓ Can not take back ownership
- ✓ No trading-cool-down mechanism

Mechanism Introduction

Buy Tax	0%
Sell Tax	0%

Sell detection

Wallets 652	Success 652	Failed 0	Siphoned 0
Tax	Ave Tax 0% Tax 0%	Count 652	

Token Holders Info

Token Holders: 5068

Top10 ratio(exclude blackhole) 18.87%

1. Cakev2: BGOAT/WBNB	61.96M (6.2%)
2.0x...1d3c	30.48M (3.05%)
3.0x...a05c	28.61M (2.86%)
4.0x...0d98	25.18M (2.52%)
5.0x...d61d	22.61M (2.26%)
6.0x...3d26	20.02M (2%)
7.0x...a130	16.79M (1.68%)
8.0x...795e	15M (1.5%)
9.0x...8874	15M (1.5%)
10.0x...80ef	15M (1.5%)

More Details

LP

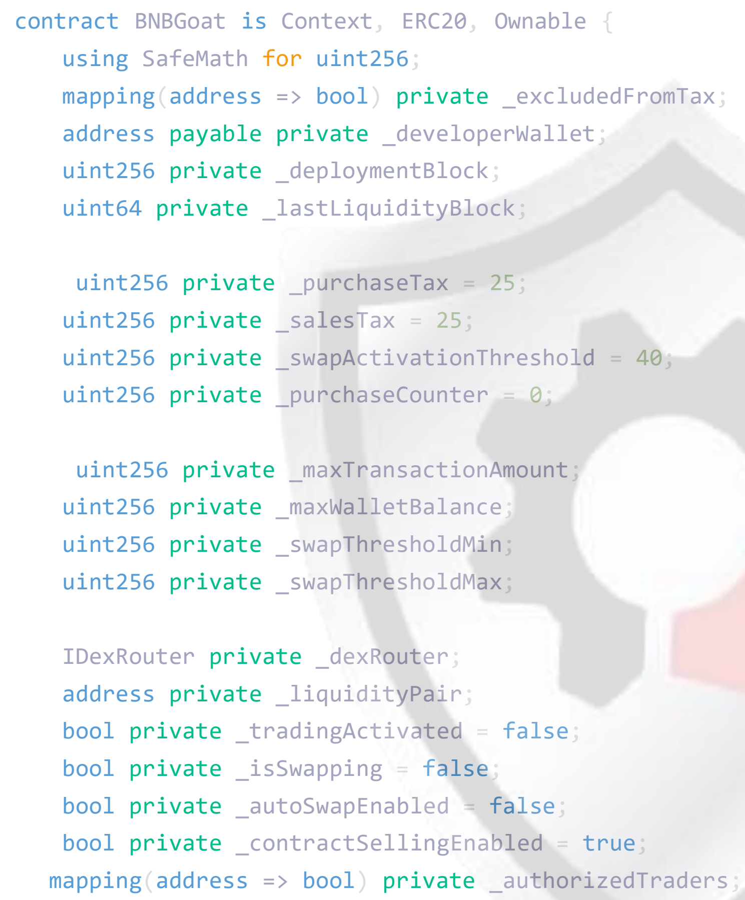
LP Holders: 10 Total Supply: 40,620.192

Percentage of LP locked 100%

1. Blackhole/黑洞地址	40.62K (100%)
2. Blackhole/黑洞地址	0.0{14}1 (0%)
3. 0x...ba7e	0 (0%)
4. 0x...71fe	0 (0%)
5. 0x...1b89	0 (0%)
6. 0x...a19c	0 (0%)
7. 0x...c7af	0 (0%)
8. 0x...dcf8	0 (0%)
9. 0x...0178	0 (0%)
10.0x...08d3	0 (0%)

More Details

Identifier	Definition	Severity
CEN-02	Initial asset distribution	Minor $ ●



```
contract BNBGoat is Context, ERC20, Ownable {
    using SafeMath for uint256;
    mapping(address => bool) private _excludedFromTax;
    address payable private _developerWallet;
    uint256 private _deploymentBlock;
    uint64 private _lastLiquidityBlock;

     uint256 private _purchaseTax = 25;
    uint256 private _salesTax = 25;
    uint256 private _swapActivationThreshold = 40;
    uint256 private _purchaseCounter = 0;

     uint256 private _maxTransactionAmount;
    uint256 private _maxWalletBalance;
    uint256 private _swapThresholdMin;
    uint256 private _swapThresholdMax;

    IDexRouter private _dexRouter;
    address private _liquidityPair;
    bool private _tradingActivated = false;
    bool private _isSwapping = false;
    bool private _autoSwapEnabled = false;
    bool private _contractSellingEnabled = true;
   mapping(address => bool) private _authorizedTraders;
```

Alleviation:

The Distribution asset was acknowledged and ultimately discarded by the **BNB GOAT**team due to Earning severity. We consider the exhibit fully attended to as it doesn't impose any meaningful security concerns.

RECOMMENDATION

Clarify intent. If anti-bot, consider using time-based or unique buyer count instead.

Contract Creator Address:

0x2B79D32B9c1309d65F5B98E7548388c961820358

Audited Files

$BGOAT.Sol

Contracts Creator Hash:

TXN HASH
0x806cc3e8c44a8ff4628cc5bd49d52ddbbf3e23f9d373df5d13f244769b28c370

Contracts:

Contract Address:
BGOAT: 0x88883075fb050f4199cF7b2011cC15f1c966403f

MANUAL REVIEW

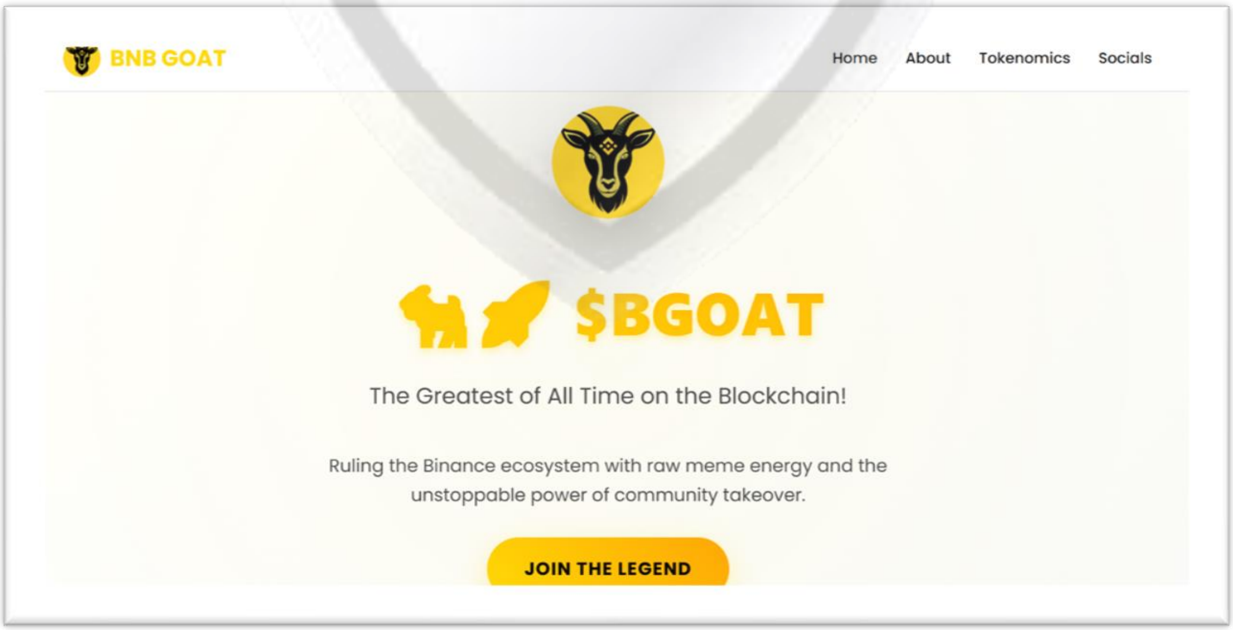
BNB GOAT:✨ is no longer just a meme token—it's a movement. What started as a spark has been claimed by a community that refuses to settle for average. This isn't hype for hype's sake; this is the kind of takeover that builds legends.

TOKEN NAME: **BNB GOAT**
Ticker: **$BGOAT**

Chain/Standard: **BINANCE NETWORK**

LAUNGUGE: **SOLIDITY**

The BNB GOAT Platform Is Launching On the Binance Network



BNB GOAT

Home About Tokenomics Socials

$BGOAT

The Greatest of All Time on the Blockchain!

Ruling the Binance ecosystem with raw meme energy and the unstoppable power of community takeover.

JOIN THE LEGEND

ISSUES CHECKING STATUS

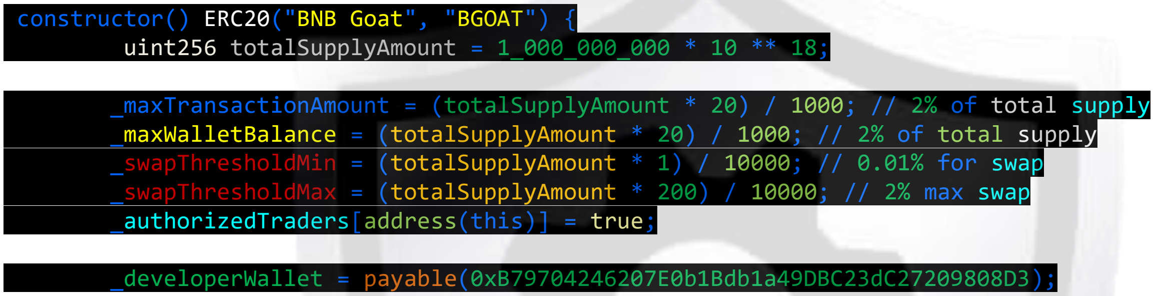
	Issue Description	Checking Status
1.	Compiler errors.	PASSED
2.	Race Conditions and reentrancy. Cross-Function Race Conditions.	PASSED
3.	Possible Delay In Data Delivery.	PASSED
4.	Oracle calls.	PASSED
5.	Front Running.	PASSED
6.	SOL Dependency.	PASSED
7.	Integer Overflow And Underflow.	PASSED
8.	DoS with Revert.	PASSED
9.	Dos With Block Gas Limit.	PASSED
10.	Methods execution permissions.	PASSED
11.	Economy Model of the contract.	PASSED
12.	The Impact Of Exchange Rate On the Move Logic.	PASSED
13.	Private use data leaks.	PASSED
14.	Malicious Event log.	PASSED
15.	Scoping and Declarations.	PASSED
16.	Uninitialized storage pointers.	PASSED
17.	Arithmetic accuracy.	PASSED
18.	Design Logic.	PASSED
19.	Cross-Function race Conditions	PASSED
20.	Save Upon Move contract Implementation and Usage.	PASSED
21.	Fallback Function Security	PASSED

AUDIT RESULT

PASSED

Identifier	Definition	Severity
CEN-02	Initial asset distribution	Minor ●

All of the initially minted assets are sent to the contract deployer when deploying the contract. This can be an issue as the deployer and/or contract owner can distribute tokens without consulting the community.



```
constructor() ERC20("BNB Goat", "BGOAT") {
        uint256 totalSupplyAmount = 1_000_000_000 * 10 ** 18;

        _maxTransactionAmount = (totalSupplyAmount * 20) / 1000; // 2% of total supply
        _maxWalletBalance = (totalSupplyAmount * 20) / 1000; // 2% of total supply
        _swapThresholdMin = (totalSupplyAmount * 1) / 10000; // 0.01% for swap
        _swapThresholdMax = (totalSupplyAmount * 200) / 10000; // 2% max swap
        _authorizedTraders[address(this)] = true;

        _developerWallet = payable(0xB79704246207E0b1Bdb1a49DBC23dC27209808D3);
```

RECOMMENDATION

Project stakeholders should be consulted during the initial asset distribution process.

RECOMMENDATION

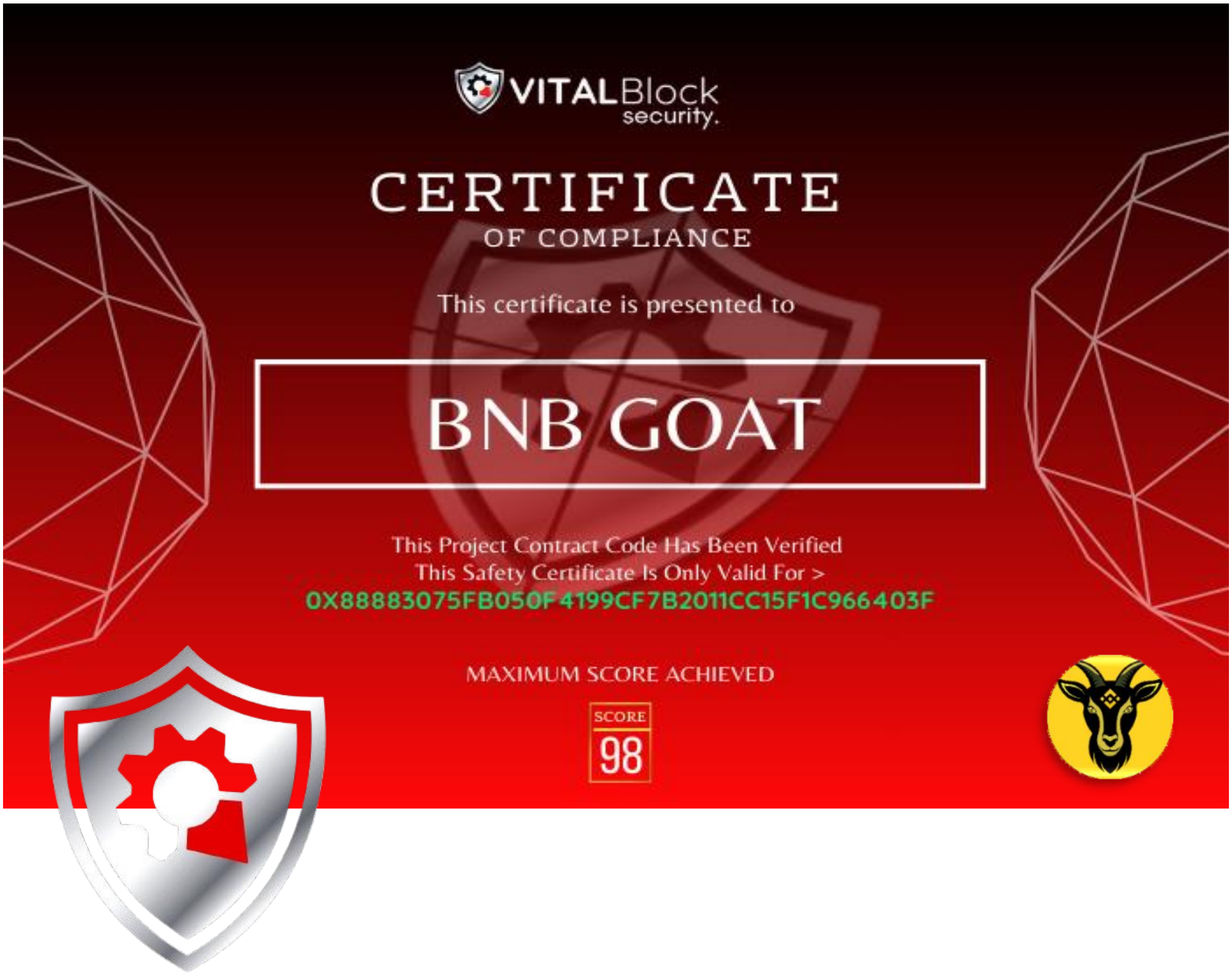
Deployer and/or contract owner private keys are secured carefully.

Please refer to PAGE-09 CENTRALIZED PRIVILEGES for a detailed understanding.

ALLEVIATION

The BNB GOAT project team understands the centralization risk. Some functions are provided privileged access to ensure a good runtime behavior in the project

CERTIFICATE BY VITAL BLOCK SECURITY



VITALBlock security.

CERTIFICATE

OF COMPLIANCE

This certificate is presented to

BNB GOAT

This Project Contract Code Has Been Verified
This Safety Certificate Is Only Valid For >
0X88883075FB050F4199CF7B2011CC15F1C966403F

MAXIMUM SCORE ACHIEVED

SCORE
98

Identifier	Definition	Severity
COD-10	Third Party Dependencies	Minor ●

Smart contract is interacting with third party protocols e.g., Pancakeswap router, cashier contract, protections contract. The scope of the audit treats third party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised, and exploited. Moreover, upgrades in third parties can create severe impacts, e.g., increased transactional fees, deprecation of previous routers, etc.

RECOMMENDATION

Inspect and validate third party dependencies regularly, and mitigate severe impacts whenever necessary.

DISCLAIMERS

Vital Block provides the easy-to-understand audit of Solidity, Move and Raw source codes (commonly known as smart contracts).

The smart contract for this particular audit was analyzed for common contract vulnerabilities, and centralization exploits. This audit report makes no statements or warranties on the security of the code. This audit report does not provide any warranty or guarantee regarding the absolute bug-free nature of the smart contract analyzed, nor do they provide any indication of the client's business, business model or legal compliance. This audit report does not extend to the compiler layer, any other areas beyond the programming language, or other programming aspects that could present security risks. Cryptographic tokens are emergent technologies, they carry high levels of technical risks and uncertainty. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. This audit report could include false positives, false negatives, and other unpredictable results.

CONFIDENTIALITY

This report is subject to the terms and conditions (including without limitations, description of services, confidentiality, disclaimer and limitation of liability) outlined in the scope of the audit provided to the client. This report should not be transmitted, disclosed, referred to, or relied upon by any individual for any purpose without InterFi Network's prior written consent.

NO FINANCIAL ADVICE

This audit report does not indicate the endorsement of any particular project or team, nor guarantees its security. No third party should rely on the reports in any way, including to make any decisions to buy or sell a product, service or any other asset. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. This audit report should not be used in any way

to make decisions around investment or involvement. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort.

FOR AVOIDANCE OF DOUBT, SERVICES, INCLUDING ANY ASSOCIATED AUDIT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

TECHNICAL DISCLAIMER

ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, VITAL BLOCK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO SERVICES, AUDIT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, VITAL BLOCK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM THE COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

WITHOUT LIMITING THE FOREGOING, VITAL BLOCK MAKES NO WARRANTY OF ANY KIND THAT ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET THE CLIENT'S OR ANY OTHER INDIVIDUAL'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE.

TIMELINESS OF CONTENT

The content contained in this audit report is subject to change without any prior notice. Vital Block does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following the publication.

LINKS TO OTHERWEBSITES

This audit report provides, through hypertext or other computer links, access to websites and social accounts operated by individuals other than Vital Block. Such hyperlinks are provided for your reference and convenience only and are the exclusive responsibility of such websites and social accounts owners. You agree that Vital block Security is not responsible for the content or operation of such websites and social accounts and that Vital Block shall have no liability to you or any other person or entity for the use of third-party websites and social accounts. You are solely responsible for determining the extent to which you may use any content at any other websites and social accounts to which you link from the report.

ABOUT VITAL BLOCK

Vital Block provides intelligent blockchain Security Solutions. We provide solidity and Raw Code Review, testing, and auditing services. We have Partnered with 15+ Crypto Launchpads, audited 50+ smart contracts, and analyzed 200,000+ code lines. We have worked on major public blockchains e.g., Ethereum, Binance, Cronos, Doge, Polygon, Avalanche, Metis, Fantom, Bitcoin Cash, Aptos, Oasis, etc.

Vital Block is Dedicated to Making Defi & Web3 A Safer Place. We are Powered by Security engineers, developers, UI experts, and blockchain enthusiasts. Our team currently consists of 5 core members, and 4+ casual contributors.

Website: https://Vitalblock.org

Email: info@vitalblock.org

GitHub: https://github.com/vital-block

Telegram (Engineering): https://t.me/vital_block

Telegram (Onboarding): https://t.me/vitalblock_cmo

Blockchain Security | Smart Contract Audit | KYC Certification | SAFU .

MADE IN CANADA

@VB_Audit

Vitalblock.org

@Vitalblock