VITALBlock security.

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

MADE IN CANADA

ViewMynte

AUDIT

SECURITY ASSESSMENT

21ST October 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	VIEWMYNTE
Methodology	**Automated Analysis, Manual Code Review**
Language	**Solidity**
Contract Address	0x88883075fb050f4199cF7b2011cC15f1c966403f
Source Code Light	**Verified**
Centralization	**Active ownership**
Compiler Version	v0.8.30+commit.73712a01
Blockchain	**BINANCE CHAIN**
Website	**https://viewmynte.com/**
Twitter	**https://x.com/viewmynte**
Telegram	**https://t.me/viewmynte_connect**
Docs	**https://viewmynte.gitbook.io/viewmynte-docs**
Prelim Report Date	**October 21ST 2025**
Final Report Date	**October 21ST 2025**

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

Document Properties

Client	VIEWMYNTE
Title	Smart Contract Audit Report
Target	VIEWMYNTE
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	October 21ST , 2025	C. John	Final Release
1.0-AP	October 21ST , 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.

- VIEWMYNTE (AS760827)
- BEP-20 Token | Address: 0x94E4510d...f6d6aB8B8 | BscScan (PPL87O54)

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 VIEWMYNTE 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.VIEWMYNTE.SOL

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	
0x94E4510dC8578F5cFb5FD47009E9236f6d6aB8B8	
Contract Name	VIEWMYNTE
Ticker	**$MYNTE**
Total Supply	**700,000,000,000**

Executive Summary

The ViewMynte contract implements a deflationary ERC-20 token with a 1%-1%-1% fee model (tax, burn, liquidity), enhanced

by a robust multisig governance system, reentrancy guards, and MEV-resistant liquidity mechanics. The code demonstrates a

high degree of security awareness, incorporating modern best practices such as custom errors, comprehensive input validation, slippage protection, and emergency controls.

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.**

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

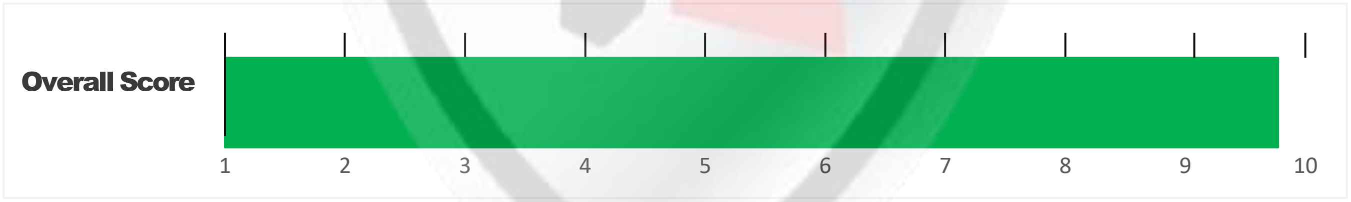
Category	Checklist Items
Basic Coding Bugs	Constructor Mismatch
	Ownership Takeover
	Redundant Fallback Function
	Overflows & Underflows
	Reentrancy
	Money-Giving Bug
	Blackhole
	Unauthorized Self-Destruct
	Revert DoS
	Unchecked External Call
	Gasless Send
	Send Instead Of Transfer
	Costly Loop
	(Unsafe) 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 VIEWMYNTE.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	0	0
Acknowledged	0	0	0	3	0
Resolved	0	0	1	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				

VIEWMYNTE Smart contract has achieved the following score: 97.0



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.

ℹ️ 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**

```
| **VIEWMYNTE** | 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❗ |
| └| MYNTE | 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 |
```

VIEWMYNTE - 01 POSSIBLE OVERFLOW

Category	Severity ●	Location	Status
Status Mathematical Operations	**Minor**	completePendingLiquidity()	**Acknowledged**

Description

In M-1: Potential Race Condition in completePendingLiquidity()

The function checks pendingLiquidity.isPending and timestamp delay, but does not validate that the pending values (tokenAmount, ethAmount) still reflect the actual contract state. If an emergency withdrawal or manual intervention occurs between _performAutoLiquidity() and completePendingLiquidity(), the contract may attempt to add liquidity with insufficient token/ETH balance, causing a revert or partial execution.

Impact: Failed liquidity addition, loss of expected LP generation, possible stuck state.

Recommendation

Before calling _addLiquidity(), re-check that:

```
  require(balanceOf(address(this)) >= tokenAmount, "Insufficient tokens");
require(address(this).balance >= ethAmount, "Insufficient ETH");
```

VIEWMYNTE - 02 POSSIBLE OVERFLOW

Category	Severity ●	Location	Status
Inconsistency	**Informational**	Constructor and manageFeeExclusion()	Acknowledged

Description

In **Inconsistent Fee Exclusion Logic for Owner**

Description:
In the constructor, msg.sender is excluded from fees. However, if ownership is transferred (via Ownable2Step), the new owner is not automatically excluded, while the old owner remains excluded. This breaks the documented assumption that “Owner should always be excluded from fees” (see manageFeeExclusion require statement).

Impact: New owner may be charged fees unintentionally, breaking expected behavior.

Recommendation

Override _transferOwnership() to:

•Remove fee exclusion from old owner (unless it’s a special address like tax wallet

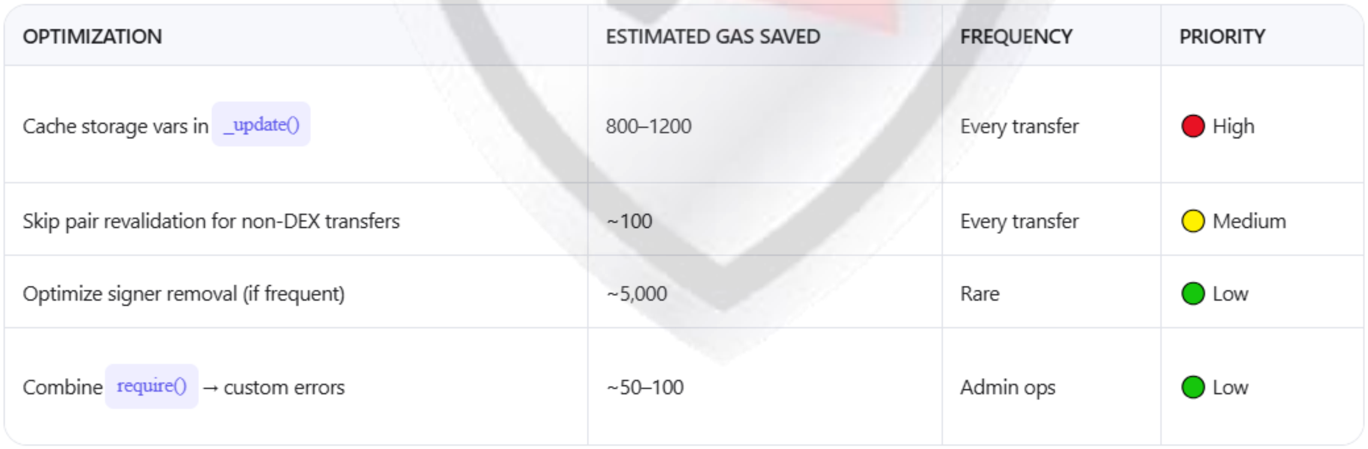
•Add fee exclusion for new owner

•Example:

```
function _transferOwnership(address newOwner) internal override {
    _excludedFromFee[owner()] = false;
    super._transferOwnership(newOwner);
    _excludedFromFee[newOwner] = true;
}
```

OPTIMIZATIONS | $MYNTE

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 ●



OPTIMIZATION	ESTIMATED GAS SAVED	FREQUENCY	PRIORITY
Cache storage vars in `_update()`	800–1200	Every transfer	🔴 High
Skip pair revalidation for non-DEX transfers	~100	Every transfer	🟡 Medium
Optimize signer removal (if frequent)	~5,000	Rare	🟢 Low
Combine `require()` → custom errors	~50–100	Admin ops	🟢 Low

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 VIEWMYNTE token can be minted by a private wallet or contract.
(Which is normal for major contract utility options)

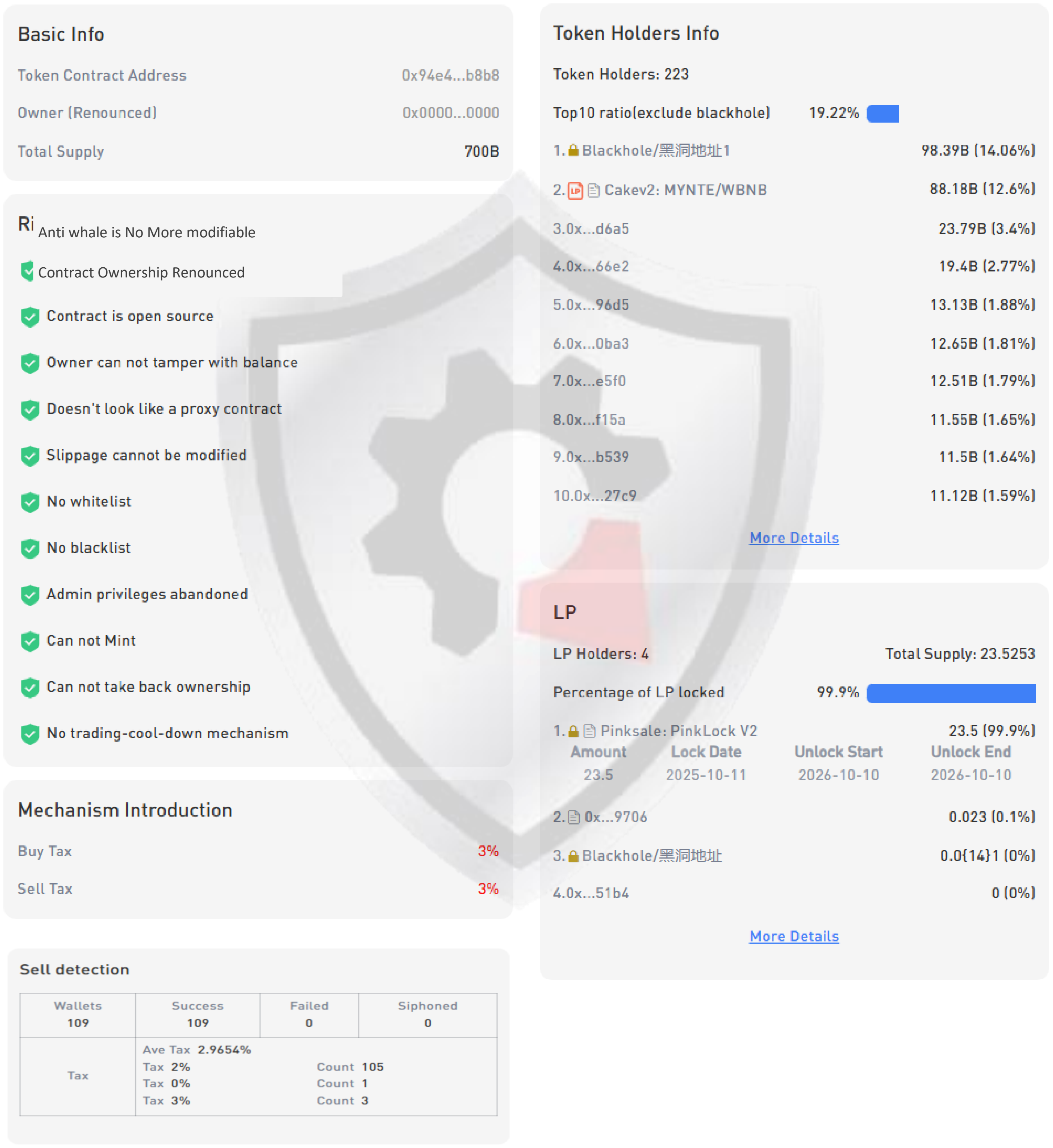
Scanning Line:

```
 uint256 totalSupplyWithDecimals = initialSupply * (10 ** decimals());
        swapTokensAtAmount = (totalSupplyWithDecimals * 10) / 10000; //
0.1%
        minimumSupply = (totalSupplyWithDecimals * 100) / 10000;    //
1%

        // FIXED: Setup exclusions BEFORE minting to prevent fees on
initial mint
        _excludedFromFee[msg.sender] = true;  // Use msg.sender instead
of owner()
        _excludedFromFee[address(this)] = true;
        _excludedFromFee[taxWallet] = true;
        _excludedFromFee[address(uniswapV2Router)] = true;

        _mint(msg.sender, totalSupplyWithDecimals);
```

Auto Contract Scan



Basic Info

Token Contract Address	0x94e4...b8b8
Owner (Renounced)	0x0000...0000
Total Supply	700B

Ri Anti whale is No More modifiable

- ✓ Contract Ownership Renounced
- ✓ 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
- ✓ Admin privileges abandoned
- ✓ Can not Mint
- ✓ Can not take back ownership
- ✓ No trading-cool-down mechanism

Mechanism Introduction

Buy Tax	3%
Sell Tax	3%

Sell detection

Wallets	Success	Failed	Siphoned
109	109	0	0
Tax	Ave Tax 2.9654%		
	Tax 2%	Count 105	
	Tax 0%	Count 1	
	Tax 3%	Count 3	

Token Holders Info

Token Holders: 223

Top10 ratio(exclude blackhole) 19.22%

1. Blackhole/黑洞地址1	98.39B (14.06%)
2. Cakev2: MYNTE/WBNB	88.18B (12.6%)
3.0x...d6a5	23.79B (3.4%)
4.0x...66e2	19.4B (2.77%)
5.0x...96d5	13.13B (1.88%)
6.0x...0ba3	12.65B (1.81%)
7.0x...e5f0	12.51B (1.79%)
8.0x...f15a	11.55B (1.65%)
9.0x...b539	11.5B (1.64%)
10.0x...27c9	11.12B (1.59%)

More Details

LP

LP Holders: 4 Total Supply: 23.5253

Percentage of LP locked 99.9%

1. Pinksale: PinkLock V2 — 23.5 (99.9%)

Amount	Lock Date	Unlock Start	Unlock End
23.5	2025-10-11	2026-10-10	2026-10-10

2. 0x...9706	0.023 (0.1%)
3. Blackhole/黑洞地址	0.0{14}1 (0%)
4.0x...51b4	0 (0%)

More Details

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

```
 function _update(address from, address to, uint256 amount) internal override {
        // FIXED: Remove zero address check that was causing constructor issues
        // The ERC20 base contract already handles zero address validation
appropriately
        require(!contractPaused, "Contract is paused");

        if (amount == 0) {
            super._update(from, to, 0);
            return;
        }

        _revalidateIfNeeded();

        uint256 contractBalance = balanceOf(address(this));
        if (contractBalance >= swapTokensAtAmount && !_inSwap && !isDEXPair[from] &&
autoLiquidityEnabled) {
            _performAutoLiquidity();
        }

        bool takeFee = !_excludedFromFee[from] &&
                       !_excludedFromFee[to] &&
                       !_inSwap &&
                       from != address(0) &&  // Not minting
                       to != address(0);       // Not burning
```

Alleviation:

The Distribution asset was acknowledged and ultimately discarded by the **VIEWMYNTE** 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.

VITALBlock security.

Contract Creator Address:

0x4821a98C98420479c514B4153EAC99AfEe7f44ff

Audited Files

$VIEWMYNTE.Sol

Contracts Creator Hash:

TXN HASH
0x6af5d4f7cc7c6b608ca8e7f4e6f4dd531e5af7cf5a08f537bc4fa510b1f0c145

Contracts:

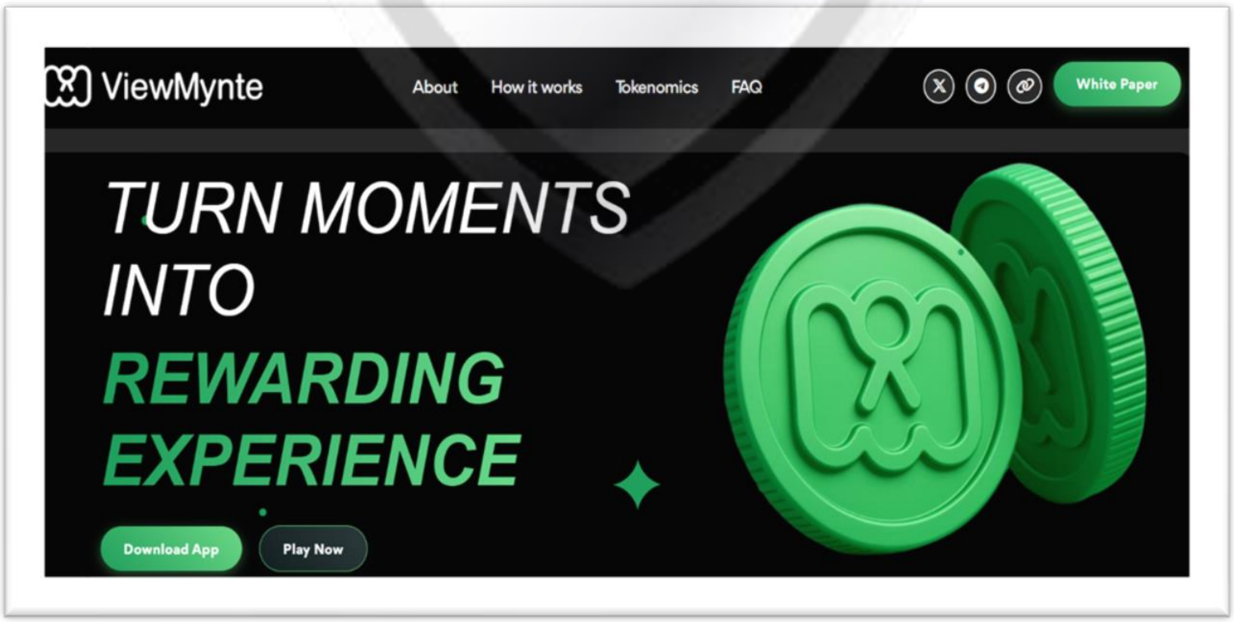
Contract Address:
MYNTE: 0x94E4510dC8578F5cFb5FD47009E9236f6d6aB8B8

MANUAL REVIEW

VIEWMYNTE: ViewMynte revolutionizes the way users interact with their mobile devices by transforming routine activities—such as streaming music, watching educational crypto videos, gaming, or even browsing social media—into opportunities to earn cryptocurrency.

TOKEN NAME: VIEWMYNTE
Ticker: $MYNTE
Chain/Standard: BINANCE NETWORK
LAUNGUGE: SOLIDITY

The VIEWMYNTE Platform Is Launching On the Binance Network



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.

```
 function completePendingLiquidity() external nonReentrant {
        require(msg.sender == pendingLiquidity.executor, "Unauthorized executor");
        require(pendingLiquidity.isPending, "No pending liquidity");
        require(block.timestamp >= pendingLiquidity.timestamp + pendingLiquidity.delay, "Delay not 
met");

        uint256 tokenAmount = pendingLiquidity.tokenAmount;
        uint256 ethAmount = pendingLiquidity.ethAmount;
```

RECOMMENDATION

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

RECOMMENDATION

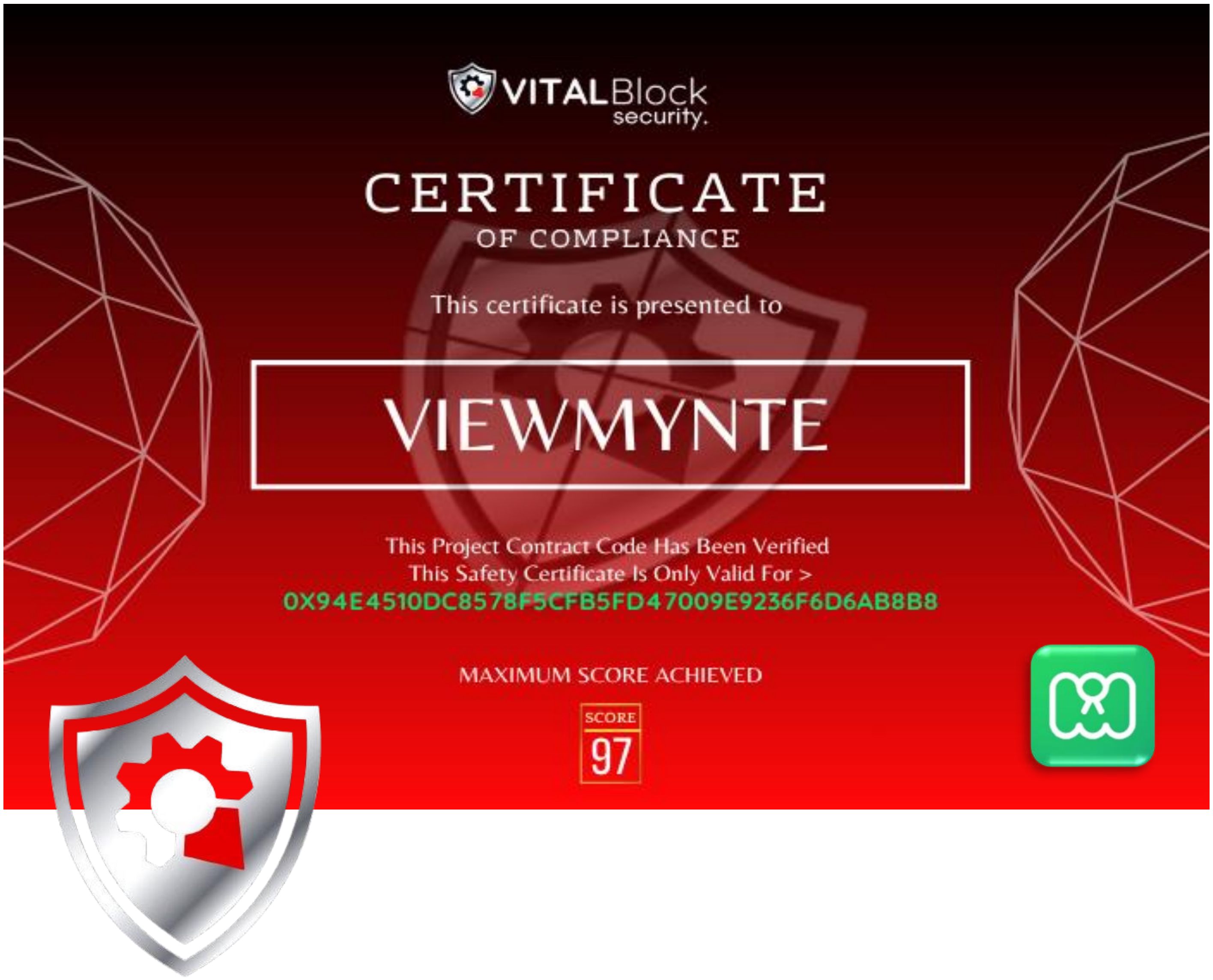
Deployer and/or contract owner private keys are secured carefully.

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

ALLEVIATION

The VIEWMYNTE 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

VIEWMYNTE

This Project Contract Code Has Been Verified

This Safety Certificate Is Only Valid For >

0X94E4510DC8578F5CFB5FD47009E9236F6D6AB8B8

MAXIMUM SCORE ACHIEVED

SCORE

97

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