VITAL Block security.

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

MADE IN CANADA

SonicSnipeBot

AUDIT

SECURITY ASSESSMENT

30th Jan 2025

For

Making Blockchain, Defi And Web3 A Safer Place.

Trail of Bits | Smart Check | MythX

@VB_Audit | Vitalblock.org | @Vitalblock

CONTENTS

INTRODUCTION

Auditing Firm	**VITAL BLOCK SECURITY**
Client Firm	**SONICSNIPEBOT**
Methodology	**Automated Analysis, Manual Code Review**
Language	**Move**
Contract Code	sui_staking_contract.move ownership.move
Source Code Light	**Open Source**
Centralization	**Active ownership**
Blockchain	**Sui Network**
Website	**https://sonicsnipebot.com**
Telegram	**https://t.me/SonicSnipePortal**
Twitter	**https://x.com/sonicSnipeBot**
Bot	**https://t.me/SonicSnipeBot**
Doc	**https://docs.sonicsnipebot.com**
Prelim Report Date	**January 28th 2025**
Final Report Date	**January 30th 2025**

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

Document Properties

Client	**SONICSNIPEBOT**
Title	Smart Contract Audit Report
Target	SONICSNIPEBOT
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	January 28TH , 2025	C. John	Final Release
1.0-AP	January 30TH , 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.

- Sonicsnipebot · Staking Contract (TNNB4550)
- https://github.com/SonicSnipeBot/SonicStakingSmartContract/blob/master/sui_staking_contract/sources/sui_staking_contract.move (ARSD522)

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 SONICSNIPEBOT to conduct the smart contract audit of its. Rust (MOVE) source code. The audit scope of work is strictly limited to the mentioned .Move file only:

`O.Move.toml`

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 File	
https://github.com/SonicSnipeBot/SonicStakingSmartContract/blob/master/sui_staking_contract/Move.toml	
Contract Name	**Sui staking contract** **OWNERSHIP.Move**
Blockchain	**Sui**

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

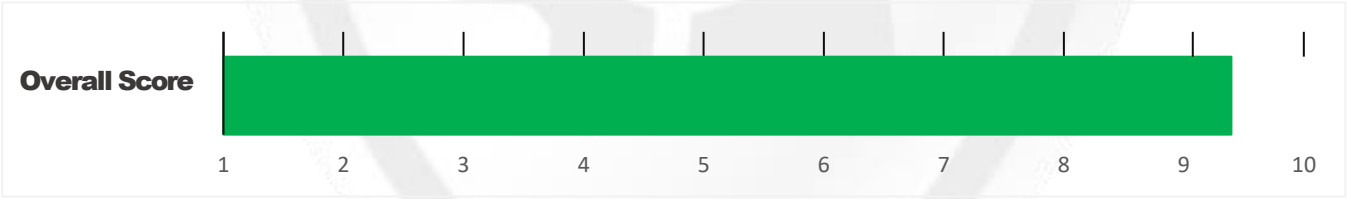
Category	Checklist Items
Basic Coding Bugs	Constructor Mismatch
	Ownership Takeover
	Redundant Fallback Function
	Overflows & Underflows
	Reentrancy
	Money-Giving Bug
	Blackhole
	Unauthorized Self-Destruct
	Revert DoS
	Unchecked External Call
	Gasless Send
	Send Instead Of Transfer
	Costly Loop
	(Unsafe) 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 SONICSNIPEBOT Move 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	0	1	0
Acknowledged	0	0	0	2	1
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				

SONICSNIPEBOT contract Code has achieved the following score: 93.0



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 is elevated.

There are some well-intended reasons have privileged 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.**

AUDIT SCOPE | SONICSNIPEBOT

ID	Repo	Comment	File	SHM321 Checksum
SSB0	contracts/sui_staking_contract.move	Cc512474	.move	788099YIRHVSK853PKTGYHHH67843OKJFGYYY766I09
SSB1	contracts/sui_staking_contract.move	cC512474	.move	347520JHDB7549H22HRTFRRE45563DESPDHBVHD655
SSB2	contracts/sui_staking_contract.move	cC512474	.move	1988Y73HUGFDINN353840OPUUYTEHHGDTFF9NNDU
SSB3	contracts/sui_staking_contract.move	cC512474	.move	4438648TEOHBF6378309EHROECNEPOEJDNETE8EYEU3
SSB4	contracts/sui_staking_contract.move	cC512474	.move	66390028765RVNKDBYFTGW553T2KOEREDW78900O7
SSB5	sui_staking_contract.move	cC512474	ownership.move	09825539BDYG543DVNKOMIKEBYRRRE4367DGVR5EUY
SSB6	sui_staking_contract.move	cC512474	ownership.move	8654RJVT3DWI865YK2643YTRFVDJBOBET8386YF3683G
SSB7	sui_staking_contract.move	cC512474	ownership.move	7763888636TGYGFFTFHBTGDCVSNDO788U59
SSB8	sui_staking_contract.move	cC512474	ownership.move	88530486494YRHFTEICBGEIEGWTWYWUHEJEHEIE33U3
SSB9	sui_token_contract_tests.move	cC512474	ownership.move	1209873KHJLKJNFJHGE98763990029774BCUHHDUU239
SSB10	sui_token_contract_tests.move	cC512474	pool.move	23456UGFYUHE98756EFHJHE7654ESDFGHGERTYUJ3897
SSB11	sui_token_contract_tests.move	cC512474	pool.move	37889UHBIONEO7TYRDFGVBN5678939IJWSFVDYUHDCI
SSB12	sui_token_contract_tests.move	cC512474	pool.move	678903098TFHJKFCPOIUGFGHJKE9865ERGBEIVBHE8767
SSB13	SonicStakingSmartContract	cC512474	Move.toml	98765SDFGBNFCOI56789UIYHGGHEJDIUYTRDCVBN3459
SSB14	SonicStakingSmartContract	cC512474	Move.toml	3348y9808hgtrusvnmu43100ejfojgfnut8496230hb574he
SSB15	SonicStakingSmartContract	cC512474	Move.toml	9864byf5f379eig28ffre64085jv1613251guhkdmue87
SSB16	SonicStakingSmartContract	cC512474	Move.toml	7ej2d8jg765tjfiowg538ij74dwftyv6478ij3gs820
SSB17	SonicStakingSmartContract	cC512474	Move.toml	864fr46de438hdguw903rfdcb246dbuhb2917enk

SSB-01 POSSIBLE OVERFLOW

Category	Severity ●	Location	Status
Status Mathematical Operations	Minor	sui_staking_contract/Move.toml	Acknowledged

Description

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

```
[addresses]
sui_staking_contract = "0x0"

# Named addresses will be accessible in Move as `@name`. They're also exported:
# for example, `std = "0x1"` is exported by the Standard Library.
# alice = "0xA11CE"
```

Address can **Not** issue more **contract staking** 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 **SonicsnipeBot** 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…

SSB-02 Key Findings

Category	Severity ●	Target	Status
Business Logic	Medium	sources/ownership.move	Low

Description

In **updateForOwner**, Relevant Function Snippet

```
public entry fun transfer_ownercap(cap: OwnerCap, to: address, ctx: &mut TxContext) {
        transfer::transfer(cap, to);
        event::emit(OwnerCapTransferredEvent {
            from: sui::tx_context::sender(ctx),
            to,
```

Description

For Ownership efficiency, the SonicSnipeBot Team is engineered with the reserve cache mechanism, which necessi-tates the common steps to be followed when operating with the reserve Ownership data in different scenarios, including the tax generation, update, and eventual persistence.

Recommendation

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

OPTIMIZATIONS | SONICSNIPEBOT

ID	Title	Category	Status
FTV	**Logarithm Refinement Optimization**	Gas Optimization	Acknowledged
FOP	**Checks Can Be Performed Earlier**	Gas Optimization	Acknowledged
FDP	**Unnecessary Use Of SafeMath**	Gas Optimization	Acknowledged
FWY	**Struct Optimization**	Gas Optimization	Acknowledged
FGT	**Unused State Variable**	Gas Optimization	Acknowledged

General Detectors

Missing Zero Address Validation

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

Attention Required

Correct Move Version

This contract uses a conventional version of Move Dependences

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

‘Z’ Pool: No additional amount of this Staking pool can be minted by a private wallet or contract…
(Which is normal for major contract utility options)

Scanning Line:

```
public entry fun update_active_status<CoinTypeA>(
        _owner_cap: &OwnerCap,
        self: &mut Pool<CoinTypeA>,
        is_active: bool,
        clock: &Clock
    ) {
        assert_version(self);
        assert!(self.is_active != is_active, E_UPDATE_ACTIVE_STATUS);

        self.is_active = is_active;
        let current_time = clock::timestamp_ms(clock) / MILLISECONDS_IN_SECONDS;
```

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

```
fun init(ctx: &mut TxContext) {
        // initialize admin cap and transfer to publisher
        transfer::transfer(OwnerCap {
            id: object::new(ctx),
        }, tx_context::sender(ctx));

        // initialize operator cap and transfer to publisher
        transfer::transfer(OperatorCap {
            id: object::new(ctx),
        }, tx_context::sender(ctx));
    }

    public entry fun transfer_ownercap(cap: OwnerCap, to: address, ctx: &mut TxContext) {
        transfer::transfer(cap, to);
        event::emit(OwnerCapTransferredEvent {
            from: sui::tx_context::sender(ctx),
            to,
```

Alleviation:

All of the initially minted assets are transfer to the contract deployer when deploying the contract. This is Normal for most deployer and/or contract owner .

RECOMMENDATION

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

Repository

https://github.com/SonicSnipeBot/SonicStakingSmartContract/

Audited Files

SONICSTAKINGCONTRACT/Move.toml
OWNERSHIP.Move

Contracts Creator Hash:

CREATOR TXN HASH:
Not Established

Contracts:

Staking Contract Address:
Not Deployed

MANUAL REVIEW

SONICSNIPEBOT: Sonic Snipe Bot is an Automated trading bot designed for fast trading execution, embedded in the Telegram Application. Sonic simplifies trading by eliminating the need to navigate to the right Dex, adjust Gas settings, or manually enter slippage.

TOKEN NAME: SONICSNIPEBOT

Chain/Standard: SUI NETWORK

LAUNGUGE: MOVE

The SONICSNIPEBOT Platform Is Launched On the Sui 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.	Move 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 Address assets are exported to the standard liabrary...

```
[addresses]
sui_staking_contract = "0x0"

# Named addresses will be accessible in Move as `@name`. They're also exported:
# for example, `std = "0x1"` is exported by the Standard Library.
# alice = "0xA11CE"
```

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

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 Security 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 OTHER WEBSITES

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