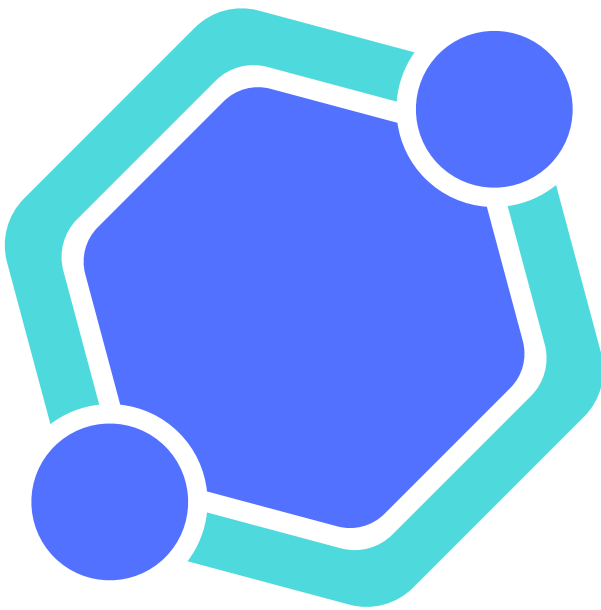


AUDIT REPORT

May 2023



Audit conducted by
RICARDO PONTES

Summary

Auditing Firm	Crypto Hub
Architecture	Crypto Hub Auditing Standard
Smart Contract Audit Approved By	Ricardo Blockchain Dev at Crypto Hub
Platform	Solidity
Mandatory Audit Check	Static, Software & Manual Analysis
Consultation Request Date	May 8, 2023
Report Date	May 8, 2023

Audit Summary

Crypto Hub team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

- ★ Mamba Coin smart contract source code has **LOW RISK SEVERITY**.
- ★ Mamba Coin has **PASSED** the smart contract audit.

For the detailed understanding of risk severity, source code vulnerability, and functional test, kindly refer to the audit.

✅ Verify the authenticity of this report on Crypto Hub Website:

<https://www.cryptohub.agency/>



Table Of Contents

Project Overview	3
Audit Scope & Methodology	4
Crypto Hub Audit Standard	5
Crypto Hub's Risk Classification	6
Smart Contract Risk Assessment	7
Contract Snapshot	7
Static / Quick Analysis	8
Software Analysis	10
SWC Attacks	13
Manual Analysis	15
Risk Status	15
Report Summary	16
Audit & KYC Certificates	17
Legal Advisory	18
Important Disclaimer	18
About Crypto Hub	19



Project Overview

Crypto Hub was consulted by Mamba Coin to conduct the smart contract security audit of their solidity source code.

Project	Mamba Coin
Blockchain	Binance Smart Chain
Language	Solidity
Contracts	0xDc6CD0621b4Ed0b344db818ea3e42d1C4C54F20C
Website:	http://mambastrikes.com/

Public logo:



Solidity Source Code On Blockchain (Verified Contract Source Code)

<https://bscscan.com/address/0xDc6CD0621b4Ed0b344db818ea3e42d1C4C54F20C#code>

Contract Name: MAMBA

Compiler Version: v0.8.18

Optimization Enabled: yes

SHA-1 Hash

Solidity source code is audited at hash

#e6cd8b15b1cdba1a54fd27dca39b10ab6c49c5d1



Audit Scope & Methodology

The scope of this report is to audit the above smart contract source code and Crypto Hub has scanned the contract and reviewed the project for common vulnerabilities, exploits, hacks, and back-doors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Smart Contract Vulnerabilities

- ☐ Re-entrancy
- ☐ Unhandled Exceptions
- ☐ Transaction Order Dependency
- ☐ Integer Overflow
- ☐ Unrestricted Action
- ☐ Incorrect Inheritance Order
- ☐ Typographical Errors
- ☐ Requirement Violation

Source Code Review

- ☐ Ownership Takeover
- ☐ Gas Limit and Loops
- ☐ Deployment Consistency
- ☐ Repository Consistency
- ☐ Data Consistency
- ☐ Token Supply Manipulation

Functional Assessment

- ☐ Access Control and Authorization
- ☐ Operations Trail and Event Generation
- ☐ Assets Manipulation
- ☐ Liquidity Access



Crypto Hub Audit Standard

The aim of Crypto Hub standard is to analyze the smart contract and identify the vulnerabilities and the hacks in the smart contract. Mentioned are the steps used by ECHELON-1 to assess the smart contract:

1. Solidity smart contract source code reviewal:

- ❖ Review of the specifications, sources, and instructions provided to Crypto Hub to make sure we understand the size, scope, and functionality of the smart contract.
- ❖ Manual review of code, which is the process of reading source code line-by-line to identify potential vulnerabilities.

2. Static, Manual, and Software analysis:

- ❖ Test coverage analysis, which is the process of determining whether the test cases are covering the code and how much code is exercised when we run those test cases.
- ❖ Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts

Automated 3P frameworks used to assess the smart contract vulnerabilities

- ❖ Slither
- ❖ Consensys MythX
- ❖ Consensus Surya
- ❖ Open Zeppelin Code Analyzer
- ❖ Solidity Code Compiler



Crypto Hub's Risk Classification

Smart contracts are generally designed to manipulate and hold funds denominated in ETH/BNB. This makes them very tempting attack targets, as a successful attack may allow the attacker to directly steal funds from the contract. Below are the typical risk levels of a smart contract: Vulnerable:

A contract is vulnerable if it has been flagged by a static analysis tool as such. As we will see later, this means that some contracts may be vulnerable because of a false-positive.

Exploitable: A contract is exploitable if it is vulnerable and the vulnerability could be exploited by an external attacker. For example, if the “vulnerability” flagged by a tool is in a function which requires to own the contract, it would be vulnerable but not exploitable.

Exploited: A contract is exploited if it received a transaction on the main network which triggered one of its vulnerabilities. Therefore, a contract can be vulnerable or even exploitable without having been exploited.

Risk severity	Meaning
! Critical	This level of vulnerability could be exploited easily, and can lead to asset loss, data loss, asset manipulation, or data manipulation. They should be fixed right away.
! High	This level vulnerabilities are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to critical risk severity
! Medium	This level of vulnerabilities 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	This level of vulnerability can be ignored. They are code style violations, and informational statements in the code. They may not affect the smart contract execution



Smart Contract Risk Assessment

Contract Snapshot

```
// [Logo]
// Created by CRYPTO HUB - WEB3 AGENCY // cryptohub.agency
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.18;

/*
 * @x10ED43C718714eb63d5aA57B78B54704E256024E
 * Details needed for deployment
 * pancakeRouter:
 * wbnb: 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
 */
/*
 * Edit these constant before deployment
 */
string constant NAME = "MAMBA COIN";

string constant SYMBOL = "MAMBA";

uint256 constant TOTAL_SUPPLY = 4000000000000000;

uint256 constant DECIMALS = 18;

// used to calculate percentages (so that we can use decimals) if you change this please change the tax
// constants below
// if in doubt leave it as is
uint256 constant FEE_DENOMINATOR = 1000;

/*
 * Percentages are calculated by dividing the tax by the FEE_DENOMINATOR
 */
uint64 constant SELL_ADMIN_TAX = 20;

// 7%
uint64 constant SELL_AUTO_LP_TAX = 10;

// 0.5%
uint64 constant SELL_CHARITY_TAX = 0;

// 0.5%
uint64 constant SELL_REFLECTION_TAX = 10;

// 1%
uint64 constant BUY_ADMIN_TAX = 20;

// 3%
uint64 constant BUY_AUTO_LP_TAX = 10;

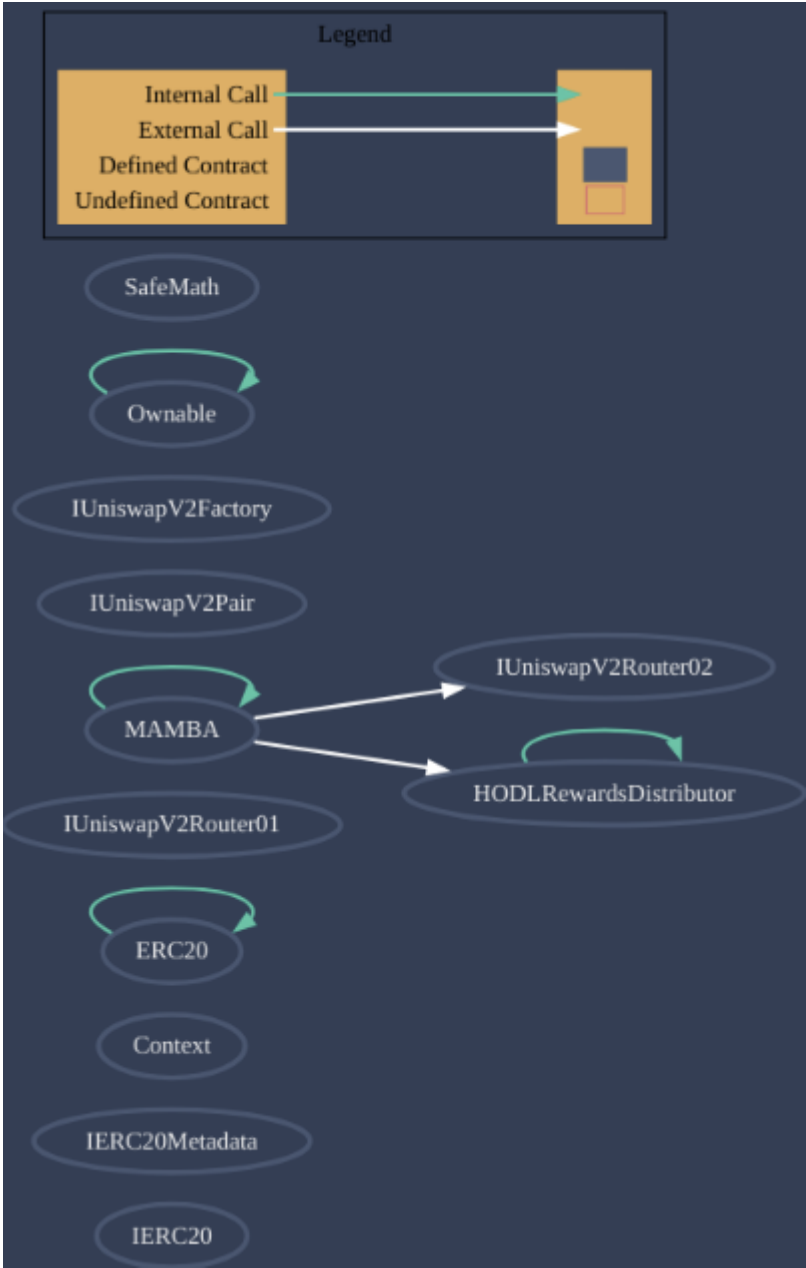
// 0.5%
uint64 constant BUY_CHARITY_TAX = 0;

// 0.5%
uint64 constant BUY_REFLECTION_TAX = 10;

// 1%
uint256 constant MAX_SELL_AMOUNT_IN_MAMBA = TOTAL_SUPPLY * 10 ** DECIMALS;
```



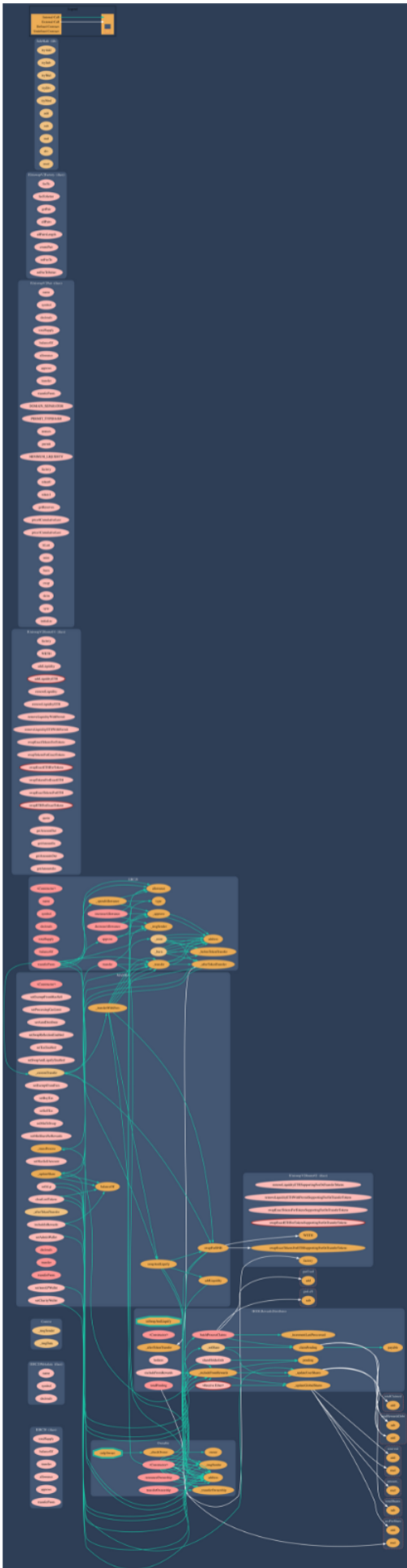
Software Analysis



Sighash	Function Signature
35082933	⇒ setAdminWallet(address)
39589351	⇒ increaseAllowance(address,uint256)
18160ddd	⇒ totalSupply()
78a88231	⇒ balanceOf(address)
a9059cbb	⇒ transfer(address,uint256)
d07e3e34	⇒ allowance(address,address)
095e87b3	⇒ approve(address,uint256)
230872dd	⇒ transferFrom(address,address,uint256)
06fddc03	⇒ name()
95d89b41	⇒ symbol()
313c25b7	⇒ decimals()
119d72af	⇒ _msgSender()
8b49d47e	⇒ _msgData()
a457c2d7	⇒ decreaseAllowance(address,uint256)
38c8789c	⇒ transfer(address,address,uint256)
4e6ec247	⇒ _mint(address,uint256)
6161eb18	⇒ burn(address,uint256)
104481f1	⇒ _approve(address,address,uint256)
1532335e	⇒ _spendAllowance(address,address,uint256)
cad3be83	⇒ _beforeTokenTransfer(address,address,uint256)
8f811a1c	⇒ _afterTokenTransfer(address,address,uint256)
c45a0150	⇒ factory()
abc4e4d0	⇒ _MINT()
e8e33700	⇒ addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
f305d719	⇒ addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
baa2abde	⇒ removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
02751c6c	⇒ removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
2199999e	⇒ removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
dwd9382a	⇒ removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
38ed1739	⇒ swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
8803d8ae	⇒ swapTokensForExactTokens(uint256,uint256,address[],address,uint256)
7ff36ab5	⇒ swapExactETHForTokens(uint256,address[],address,uint256)
4a25d94a	⇒ swapTokensForExactETH(uint256,uint256,address[],address,uint256)
10c8af65	⇒ swapExactTokensForETH(uint256,uint256,address[],address,uint256)
1b3dd4d1	⇒ swapETHForExactTokens(uint256,address[],address,uint256)
a615d0c	⇒ quote(uint256,uint256,uint256)
054958d4	⇒ getAmountOut(uint256,uint256,uint256)
85f8c259	⇒ getAmountIn(uint256,uint256,uint256)
d86c081f	⇒ getAmountsOut(uint256,address[])
1f08ca74	⇒ getAmountsIn(uint256,address[])
e72979ab	⇒ removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256)
5b0d5984	⇒ removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
5c11d795	⇒ swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
b6f9de95	⇒ swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)
791ac947	⇒ swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
364ae15	⇒ PERMIT_SUPPORT_TOKEN()
38adff1f	⇒ PERMIT_TYPEHASH()
7ecebe00	⇒ nonces(address)
d505accf	⇒ permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
ba9a7a56	⇒ MINIMUM_LIQUIDITY()
00f16891	⇒ token()
d12128a7	⇒ token1()
090711ac	⇒ getReserves()
5909c0d5	⇒ priceCumulativeLast()
5a3f5493	⇒ priceCumulativeLast()
7464fcd3d	⇒ kLast()
6a627842	⇒ kint(address)
80afcb44	⇒ burn(address)
07210d9f	⇒ swap(uint256,uint256,address,bytes)
bc25cf77	⇒ skim(address)
ff6c0ae9	⇒ sync()
485cc955	⇒ initialize(address,address)
017c7e56	⇒ feeTo()
094b7415	⇒ feeToSetter()
e6ad3985	⇒ getPair(address,address)
1e3dd18b	⇒ allPairs(uint256)
57412ba3	⇒ allPairLength()
cb453396	⇒ createPair(address,address)
f40901ed	⇒ setFeeTo(address)
a2c74af6	⇒ setFeeToSetter(address)
8da5cb5b	⇒ owner()
53a72975	⇒ _checkOwner()
715918a6	⇒ renounceOwnership()
f2f4e380	⇒ transferOwnership(address)
d29d44ee	⇒ transferOwnership(address)
884557bf	⇒ tryAdd(uint256,uint256)
a29962b1	⇒ trySub(uint256,uint256)
6281ef4d	⇒ tryMul(uint256,uint256)
736eca18	⇒ tryDiv(uint256,uint256)
38d08067	⇒ tryMod(uint256,uint256)
771602f7	⇒ add(uint256,uint256)
b67d77c5	⇒ sub(uint256,uint256)
cb4a49c	⇒ mul(uint256,uint256)
a91c15b	⇒ div(uint256,uint256)
f43f523a	⇒ mod(uint256,uint256)
e31bdc0a	⇒ sub(uint256,uint256,string)
b745d336	⇒ div(uint256,uint256,string)
71af23a8	⇒ mod(uint256,uint256,string)
5ecb0a20	⇒ pending(address)
319907ba	⇒ totalPending()
14d18a3b	⇒ shareholderInfo(address)
8188f71c	⇒ holders()
878959e4	⇒ setShare(address,uint256)
111e0376	⇒ excludeFromRewards(address)
c0b070a8	⇒ excludeFromRewards(address)
0d817183	⇒ claimPending(address)
92a0c152	⇒ batchProcessClaims(uint256)
46f2201e	⇒ _updateGlobalShares(uint256)
05c3bfff1	⇒ _updateUserShares(address,uint256)
dc1b08e9	⇒ _incrementLastProcessed()
e10a4180	⇒ setExemptFromBail(address,bool)
d31fab1c	⇒ setProcessingGasLimit(uint256)
4e14ec4b	⇒ setAutoDistribute(bool)
1c44ab31	⇒ setSwapReflectionEnabled(bool)
cbaf58b0	⇒ setTaxEnabled(bool)
c49b9e08	⇒ setSwapAndLiquifyEnabled(bool)
4b7aeb31	⇒ setSLP(address,bool)
77627d11	⇒ setExemptFromFees(address,bool)
a6919b5	⇒ setBuyTax(uint64,uint64,uint64,uint64)
e4ae7f3a	⇒ setSellTax(uint64,uint64,uint64,uint64)
ac1d05a9	⇒ setMinSwap(uint256)
c9253800	⇒ setMinShareRewards(uint256)
8ca356a6	⇒ customTransfer(address,address,uint256)
a3b26f33	⇒ transferWithFees(address,address,uint256)
e99c9d09	⇒ setMaxSellAmount(uint256)
a5cbe7f1	⇒ setAutoPwallet(address)
385b30d7	⇒ setCharityWallet(address)
173865ad	⇒ swapAndLiquify(uint256)
83836efa	⇒ swapForBNB(uint256)
9cd441da	⇒ addLiquidity(uint256,uint256)
a24bf377	⇒ _massProcess()
574b0802	⇒ updateShare(address)
85d2c277	⇒ clearTokens(address,address,uint256)
b609995e	⇒ includeInRewards(address)







SWC Attacks

The following table contains an overview of the SWC registry. Each row consists of an SWC identifier (ID), weakness title, CWE parent and list of related code samples.

The auditor used a MythX tool, A static analyzer that parses the Solidity AST, a symbolic analyzer that detects possible vulnerable states, and a greybox fuzzer that detects vulnerable execution paths.

ID	Description	Status
SWC - 100	Function Default Visibility	✓ Passed
SWC - 101	Integer Overflow and Underflow	✓ Passed
SWC - 102	Outdated Compiler Version	✓ Passed
SWC - 103	Floating Pragma	✓ Passed
SWC - 104	Unchecked Call Return Value	✓ Passed
SWC - 105	Unprotected Ether Withdrawal	✓ Passed
SWC - 106	Unprotected SELFDESTRUCT Instruction	✓ Passed
SWC - 107	Reentrancy Passed	✓ Passed
SWC - 108	State Variable Default Visibility	✓ Passed
SWC - 109	Uninitialized Storage Pointer	✓ Passed
SWC - 110	Assert Violation Passed	✓ Passed
SWC - 111	Use of Deprecated Solidity Functions	✓ Passed
SWC - 112	Delegatecall to Untrusted Callee	✓ Passed
SWC - 113	DoS with Failed Call	✓ Passed
SWC - 114	Transaction Order Dependence	✓ Passed
SWC - 115	Authorization through tx.origin	✓ Passed
SWC - 116	Block values as a proxy for time	✓ Passed
SWC - 117	Signature Malleability	✓ Passed



ID	Description	Status
SWC - 118	Incorrect Constructor Name	✓ Passed

SWC - 119	Shadowing State Variables	✓ Passed
SWC - 120	Weak Sources of Randomness from Chain Attributes	✓ Passed
SWC - 121	Missing Protection against Signature Replay Attacks	✓ Passed
SWC - 122	Lack of Proper Signature Verification	✓ Passed
SWC - 123	Requirement Violation Passed	✓ Passed
SWC - 124	Write to Arbitrary Storage Location	✓ Passed
SWC - 125	Incorrect Inheritance Order Passed	✓ Passed
SWC - 126	Insufficient Gas Griefing	✓ Passed
SWC - 127	Arbitrary Jump with Function Type Variable	✓ Passed
SWC - 128	DoS With Block Gas Limit	✓ Passed
SWC - 129	Typographical Error	✓ Passed
SWC - 130	Right-To-Left-Override control character (U+202E)	✓ Passed
SWC - 131	Presence of unused variables	✓ Passed
SWC - 132	Unexpected Ether balance	✓ Passed
SWC - 133	Hash Collisions With Multiple Variable Arguments	✓ Passed

SWC - 134	Message call with hardcoded gas amount	✓ Passed
SWC - 135	Code With No Effects	✓ Passed
SWC - 136	Unencrypted Private Data On-Chain	✓ Passed



Manual Analysis

No Risks Detected

Risk Status

Risk severity	Meaning
! Critical	None critical severity issues identified
! High	None high severity issues identified
! Medium	None medium severity issues identified
! Low	None low severity issues identified
Verified	27 functions and instances verified and checked
Safety Score	91 out of 100

Report Summary

Crypto Hub team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks.

MAMBA smart contract source code has **LOW RISK SEVERITY**.

MAMBA has **PASSED** the smart contract audit.



Note for stakeholders:

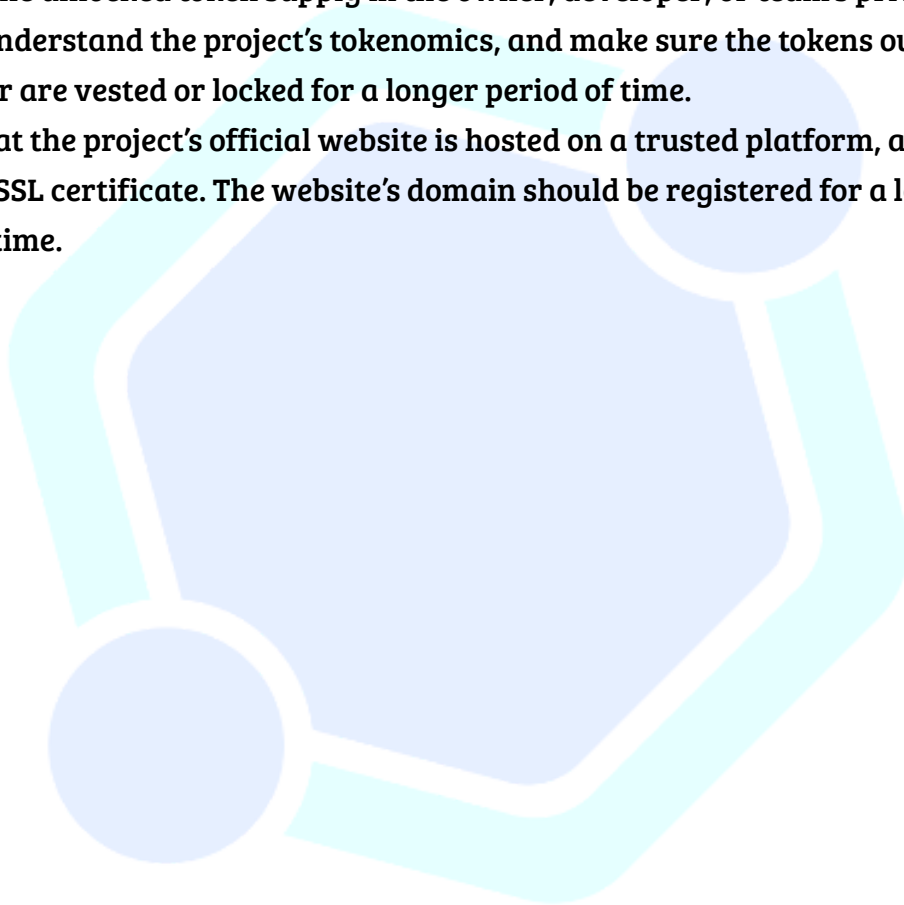
Be aware that active smart contract owner privileges constitute an elevated impact on smart contract's safety and security.

Make sure that the project team's KYC/identity is verified by an independent firm, e.g., Crypto Hub.

Always check if the contract's liquidity is locked. A longer liquidity lock plays an important role in the project's longevity. It is recommended to have multiple liquidity providers.

Examine the unlocked token supply in the owner, developer, or team's private wallets. Understand the project's tokenomics, and make sure the tokens outside of the LP Pair are vested or locked for a longer period of time.

Ensure that the project's official website is hosted on a trusted platform, and is using an active SSL certificate. The website's domain should be registered for a longer period of time.



Audit & KYC Certificates

We hereby certificate MAMBA token smart contract as an audited project under the Crypto Hub enterprise umbrella. And to represent it as such we issued the following certificate:



Legal Advisory

Important Disclaimer

Crypto Hub provides contract auditing and project verification services for blockchain projects. The purpose of the audit is to analyze the on-chain smart contract source code, and to provide a basic overview of the project. This report should not be transmitted, disclosed, referred to, or relied upon by any person for any purposes without Crypto Hub prior written consent.

Crypto Hub provides the easy-to-understand assessment of the project, and the smart contract (otherwise known as the source code). The audit makes no statements or warranties on the security of the code. It also cannot be considered as an adequate assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have used all the data at our disposal to provide the transparent analysis, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Be aware that smart contracts deployed on a blockchain aren't resistant from external vulnerability, or a hack. Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security. Therefore, Crypto Hub does not guarantee the explicit security of the audited smart contract.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

This report should not be considered as an endorsement or disapproval of any project or team. 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. Do conduct your own due diligence and consult your financial advisor before making any investment decisions.



About Crypto Hub

Crypto Hub provides intelligent blockchain solutions. Crypto Hub is developing an ecosystem that is seamless and responsive. Some of our services: Blockchain Security, Token Launchpad, NFT Marketplace, etc. Crypto Hub's mission is to interconnect multiple services like Blockchain Security, DeFi, Gaming, and Marketplace under one ecosystem that is seamless, multi-chain compatible, scalable, secure, fast, responsive, and easy-to-use.

Crypto Hub is built by a decentralized team of UI experts, contributors, engineers, and enthusiasts from all over the world. Our team currently consists of 3+ core team members, and 6+ casual contributors. Crypto Hub provides manual, static, and automatic smart contract analysis, to ensure that the project is checked against known attacks and potential vulnerabilities.



