CFG NINJA AUDITS

Security Assessment

MILESTONEMILLIONS Token

September 13, 2023

Audit Status: Pass

Audit Edition: Advance - Simulation

Milestone Millions

POWERED BY
BLADE POOL

Risk Analysis

Classifications of Manual Risk Results

Classification	Description
🔴 Critical	Danger or Potential Problems.
🟠 High	Be Careful or Fail test.
🟢 Low	Pass, Not-Detected or Safe Item.
ℹ Informational	Function Detected

Manual Code Review Risk Results

Contract Priviledge	Description
🟢 Buy Tax	3%
🟢 Sale Tax	10%
🟢 Cannot Sale	Pass
🟢 Cannot Sale	Pass
🟢 Max Tax	10%
🟡 Modify Tax	Yes
🟢 Fee Check	Pass
🟢 Is Honeypot?	Not Detected
🟢 Trading Cooldown	Not Detected
🟢 Can Pause Trade?	Pass

Contract Priviledge	Description
🟡 Pause Transfer?	Detected - Currently Enable
🔴 Max Tx?	Fail
🔴 Is Anti Whale?	Detected
🟢 Is Anti Bot?	Not Detected
🟢 Is Blacklist?	Not Detected
🟢 Blacklist Check	Pass
🟢 is Whitelist?	Not Detected
🟢 Can Mint?	Pass
🟢 Is Proxy?	Not Detected
🟢 Can Take Ownership?	Not Detected
🟢 Hidden Owner?	Not Detected
ℹ Owner	0xd6af0db215b9df43a98311c19a0e213097888cef
🟢 Self Destruct?	Not Detected
🟢 External Call?	Not Detected
🟢 Other?	Not Detected
🟢 Holders	1
🟡 Auditor Confidence	Medium - Risk

The following quick summary it's added to the project overview; however, there are more details about the audit and its results. Please read every detail.

Project Overview

Token Summary

Parameter	Result
Address	0xAA72d86210AC33BcA2de6139403F9AF37398E721
Name	MILESTONEMILLIONS
Token Tracker	MILESTONEMILLIONS (MSMIL)
Decimals	18
Supply	500,000,000
Platform	Bitrock
compiler	v0.8.19+commit.7dd6d404
Contract Name	MilestoneMillions
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://scan.bit-rock.io/address/0xAA72d86210AC33BcA2de6139403F9AF37398E721/contracts#address-tabs
Payment Tx	0x

Project Overview

Simulation Summary

Parameter	Result
Transfer From Owner	Pass
Transfer From Holder	Pass
Add Liquidity	Pass
RemoveLiquidity	Pass
Buy from Owner	Pass
Buy from Holder	Pass
Sale from Owner	Pass
Sale from Holder	Pass
Remove Liquidity	Pass
SwapAndLiquify	Pass
SwapAndSale w/Fee	Pass
SwapAndSale TX	https://testnet.bscscan.com/tx/0x1ec0433ae2cd6cdf1f700848426a52d3e25d6ab83a288246e68320f2a4953d6c
SwapAndSaleNoFee	Pass
SwapAndSale No/Fee TX	https://testnet.bscscan.com/tx/0xa396a4e596cfaf3f44938eedee5831f624897696bfb90813efe9f252f31164af

Parameter	Result
ExcludeFromFees	Pass
LaunchPad	None
Pool Creation	N/A
Pool Creation TX	
Pool Finalize	N/A
Pool Finalize TX	
Enable	Pass

The following quick summary it's added to the project overview; however, there are more details about the audit and its results. Please read every detail.

Main Contract Assessed
Contract Name

Name	Contract	Live
MILESTONEMILLIONS	0xAA72d86210AC33BcA2de6139403F9AF37398E721	Yes

TestNet Contract Assessed
Contract Name

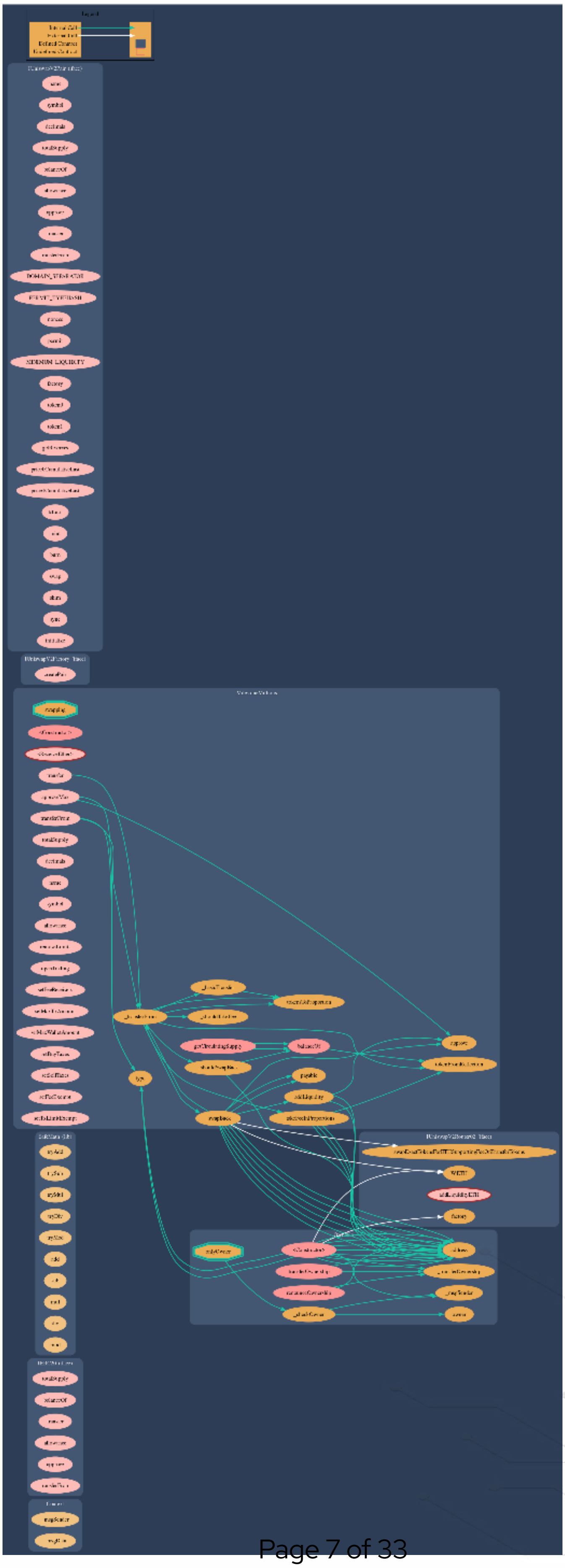
Name	Contract	Live
MILESTONEMILLIONS	0xe70985F86b79D60139be95AEA33024aC1ED93C4d	Yes

Solidity Code Provided

SolID	File Sha-1	FileName
MSMIL	0f7231790cc6093e147682eb481c5ffeea2911b0	milestonemillionsfinal.sol
MSMIL		
MSMIL		
MSMIL		
MSMIL		
MSMIL		

Call Graph

The contract for MILESTONEMILLIONS has the following call graph structure.



Smart Contract Vulnerability Checks

The Smart Contract Weakness Classification Registry (SWC Registry) is an implementation of the weakness classification scheme proposed in EIP-1470. It is loosely aligned to the terminologies and structure used in the Common Weakness Enumeration (CWE) while overlaying a wide range of weakness variants that are specific to smart contracts.

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	milestonemillionsfinal.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-103	Pass	A floating pragma is set.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-104	Pass	Unchecked Call Return Value.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	milestonemillionsfinal.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	milestonemillionsfinal.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-108	Low	State variable visibility is not set..	milestonemillionsfinal.sol	L: 562 C: 11, L: 563 C: 11,L: 565 C: 12,L: 573 C: 52,L: 577 C: 29,L: 578 C: 29,L: 605 C: 9
SWC-109	Pass	Uninitialized Storage Pointer.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-115	Low	Authorization through tx.origin.	milestonemillionsfinal.sol	L: 626 C: 35,L: 631 C: 24,L: 632 C: 20,L: 638 C: 16,L: 639 C: 34
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	milestonemillionsfinal.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-118	Pass	Incorrect Constructor Name.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randonmness.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U +202E).	milestonemillionsfinal.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	milestonemillionsfinal.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-132	Pass	Unexpected Ether balance.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	milestonemillionsfinal.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	milestonemillionsfinal.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	milestonemillionsfinal.sol	L: 0 C: 0

We scan the contract for additional security issues using MYTHX and industry-standard security scanning tools.

Smart Contract Vulnerability Details

SWC-108 - State Variable Default Visibility

CWE-710: Improper Adherence to Coding Standards

Description:

Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

Remediation:

Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.

References:

Ethereum Smart Contract Best Practices - Explicitly mark visibility in functions and state variables

Smart Contract Vulnerability Details

SWC-115 - Authorization through tx.origin

CWE-477: Use of Obsolete Function

Description:

tx.origin is a global variable in Solidity which returns the address of the account that sent the transaction. Using the variable for authorization could make a contract vulnerable if an authorized account calls into a malicious contract. A call could be made to the vulnerable contract that passes the authorization check since tx.origin returns the original sender of the transaction which in this case is the authorized account.

Remediation:

tx.origin should not be used for authorization. Use msg.sender instead.

References:

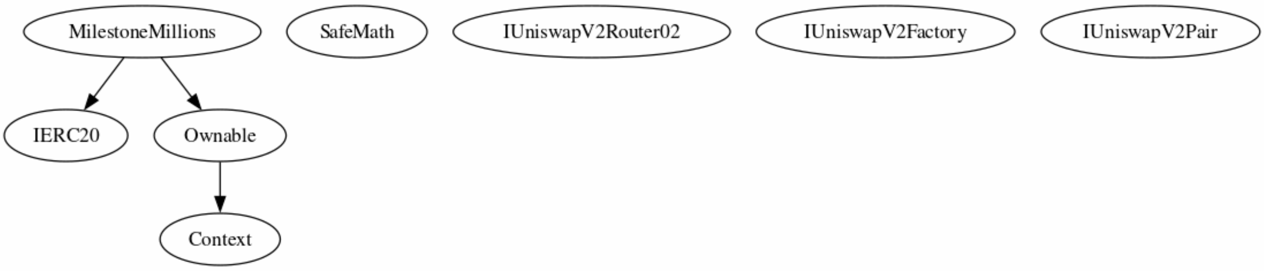
Solidity Documentation - tx.origin

Ethereum Smart Contract Best Practices - Avoid using tx.origin

SigmaPrime - Visibility.

Inheritance

The contract for MILESTONEMILLIONS has the following inheritance structure.



Smart Contract Advance Checks

ID	Severity	Name	Result	Status
MSMIL-01	Low	Potential Sandwich Attacks.	Pass	Not-Found
MSMIL-02	Informational	Function Visibility Optimization	Fail	Detected
MSMIL-03	Low	Lack of Input Validation.	Fail	Detected
MSMIL-04	High	Centralized Risk In addLiquidity.	Fail	Detected
MSMIL-05	Low	Missing Event Emission.	Fail	Detected
MSMIL-06	Low	Conformance with Solidity Naming Conventions.	Pass	Not Detected
MSMIL-07	Low	State Variables could be Declared Constant.	Fail	Detected
MSMIL-08	Low	Dead Code Elimination.	Pass	Not Detected
MSMIL-09	High	Third Party Dependencies.	Pass	Not Detected
MSMIL-10	High	Initial Token Distribution.	Pass	Not Detected
MSMIL-11	Medium	Unable to update swapThreshold within the contract.	Fail	Detected
MSMIL-12	High	Centralization Risks In The X Role	Pass	Not Detected
MSMIL-13	Informational	Extra Gas Cost For User..	Fail	Detected
MSMIL-14	Medium	Unnecessary Use Of SafeMath	Fail	Detected

ID	Severity	Name	Result	Status
MSMIL-15	Medium	Symbol Length Limitation due to Solidity Naming Standards.	Pass	Not Detected
MSMIL-16	Medium	Taxes can be up to 100%	Pass	Not Detected
MSMIL-17	Logical Issue	Highly Permissive Role Access.,`	Pass	Not Detected
MSMIL-18	Critical	Stop Transactions by using Enable Trade.	Pass	Resolved

MSMIL-02 | Function Visibility Optimization.

Category	Severity	Location	Status
Gas Optimization	ⓘ Informational	milestonemillionsfinal.sol: L: 562 C: 11, L: 563 C: 11,L: 565 C: 12,L: 573 C: 52,L: 577 C: 29,L: 578 C: 29,L: 605 C: 9	Detected

Description

The following functions are declared as public and are not invoked in any of the contracts contained within the projects scope:

Function Name	Parameters	Visibility
_name		internal
_symbol		internal
_totalSupply		internal
_allowances		internal
isFeeExempt		internal
isTxLimitExempt		internal
inSwap		internal

The functions that are never called internally within the contract should have external visibility

Remediation

We advise that the function's visibility specifiers are set to external, and the array-based arguments change their data location from memory to calldata, optimizing the gas cost of the function.

References:

external vs public best practices.

MSMIL-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	● Low	milestonemillionsfinal.sol: L: 811 C: 14, L: 806 C: 14, L: 806 C: 14,L: 745 C: 14	Detected

Description

The given input is missing the check for the non-zero address.

The given input is missing the check for the all onlyOwners need to have a required functions..

Remediation

We advise the client to add the check for the passed-in values to prevent unexpected errors as below:

```
    ...
     require(receiver != address(0), "Receiver is the zero address");
    ...
    ...
    require(value X limitation, "Your not able to do this function");
   ...
```

We also recommend customer to review the following function that is missing a required validation. all onlyOwners need to have a required functions..

MSMIL-04 | Centralized Risk In addLiquidity.

Category	Severity	Location	Status
Coding Style	● High	milestonemillionsfinal.sol: L: 937 C: 14	Detected

Description

uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this), tokenAmount, 0, 0, owner(), block.timestamp);

The addLiquidity function calls the uniswapV2Router.addLiquidityETH function with the to address specified as owner() for acquiring the generated LP tokens from the MSMIL-WBNB pool.

As a result, over time the _owner address will accumulate a significant portion of LP tokens.If the _owner is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Remediation

We advise the to address of the uniswapV2Router.addLiquidityETH function call to be replaced by the contract itself, i.e. address(this) , and to restrict the management of the LP tokens within the scope of the contract's business logic. This will also protect the LP tokens from being stolen if the _owner account is compromised. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

1. Indicatively, here are some feasible solutions that would also mitigate the potential risk:
2. Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
3. Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;

Introduction of a DAO / governance / voting module to increase transparency and user involvement

Project Action

MSMIL-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	● Low	milestonemillionsfinal.sol: L: 811 C: 14, L: 806 C: 14,L: 787 C: 14,L: 768 C: 14,L: 762 C: 14,L: 756 C: 14,L: 745 C: 14,L: 734 C: 14	Detected

Description

Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.The linked code does not create an event for the transfer.

Remediation

Emit an event for critical parameter changes. It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

MSMIL-07 | State Variables could be Declared Constant.

Category	Severity	Location	Status
Coding Style	🟢 Low	milestonemillionsfinal.sol: L: 562 C: 11, L: 563 C: 11	Detected

Description

Constant state variables should be declared constant to save gas.

```
_name
_symbol
```

Remediation

Add the constant attribute to state variables that never changes.

https://docs.soliditylang.org/en/latest/contracts.html#constant-state-variables

MSMIL-11 | Unable to update swapThreshold within the contract..

Category	Severity	Location	Status
Optimization	● Medium	milestonemillionsfinal.sol: L: 604 C: 14	Detected

Description

The Contract defines the following value, uint256 public swapThreshold = (_totalSupply * 1) / 1000;. However the following value cannot be modified, is important to ensure the value can be changed to avoid price impact as token price increase or decrease.

Remediation

Create a external onlyOwner function to update swapThreshold, this will ensure that the swap tokens can be controlled at any given time.

Project Action

MSMIL-13 | Extra Gas Cost For User.

Category	Severity	Location	Status
Logical Issue	Informational	milestonemillionsfinal.sol: L: 845, C: 0	Detected

Description

The user may trigger a tax distribution during the transfer process, which will cost a lot of gas and it is unfair to let a single user bear it.

Remediation

We advise the client to make the owner responsible for the gas costs of the tax distribution.

Project Action

_swapBack();

MSMIL-14 | Unnecessary Use Of SafeMath

Category	Severity	Location	Status
Logical Issue	● Medium	milestonemillionsfinal.sol: L: 96 C: 14	Detected

Description

The SafeMath library is used unnecessarily. With Solidity compiler versions 0.8.0 or newer, arithmetic operations
will automatically revert in case of integer overflow or underflow.
library SafeMath {
An implementation of SafeMath library is found.
using SafeMath for uint256;
SafeMath library is used for uint256 type in contract.

Remediation

We advise removing the usage of SafeMath library and using the built-in arithmetic operations provided by the
Solidity programming language

Project Action

Technical Findings Summary

Classification of Risk

Severity	Description
🔴 Critical	Risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
🟠 High	Risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
🟡 Medium	Risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform
🟢 Low	Risks can be any of the above but on a smaller scale. They generally do not compromise the overall integrity of the Project, but they may be less efficient than other solutions.
ℹ Informational	Errors are often recommended to improve the code's style or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

Findings

Severity	Found	Pending	Resolved
🔴 Critical	0	0	1
🟠 High	1	0	0
🟡 Medium	2	0	0
🟢 Low	3	0	0
ℹ Informational	2	0	0
Total	8	0	0

Social Media Checks

Social Media	URL	Result
Twitter	https://www.Twitter.com/milestnmillions	Pass
Other	https://t.me/milestonemillionsannouncements	Pass
Website	https://www.milestonemillions.com	Pass
Telegram	https://t.me/milestonemillions	Pass

We recommend to have 3 or more social media sources including a completed working websites.

Social Media Information Notes:

Auditor Notes: undefined

Project Owner Notes:

Assessment Results

Score Results

Review	Score
Overall Score	80/100
Auditor Score	80/100

Review by Section	Score
Manual Scan Score	28/33
SWC Scan Score	33 /37
Advance Check Score	19 /30

The Following Score System Has been Added to this page to help understand the value of the audit, the maximun score is 100, however to attain that value the project most pass and provide all the data needed for the assessment. Our Passing Score has been changed to 80 Points, if a project does not attain 80% is an automatic failure. Read our notes and final assessment below.

Audit Passed

Assessment Results

Important Notes:

- Simulation performed on BSC Testnet.
- The customer deployed the mainnet contract to bitrock, is important to review the bitrock documentation.
- Please DYOR on the project.

Auditor Score =80
Audit Passed

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that actagainst the nature of decentralization, such as explicit ownership or specialized access roles incombination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimalEVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on howblock.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functionsbeing invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that mayresult in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to makethe codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code,such as a constructor assignment imposing different require statements on the input variables than a setterfunction.

Coding Best Practices

ERC 20 Conding Standards are a set of rules that each developer should follow to ensure the code meet a set of creterias and is readable by all the developers.

Disclaimer

CFGNINJA has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocation for the Project, and users relying on this report should not consider this as having any merit for financial advice in any shape, form, or nature. The contracts audited do not account for any economic developments that the Project in question may pursue, and the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude, and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are entirely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence, regardless of the findings presented. Information is provided 'as is, and CFGNINJA is under no covenant to audited completeness, accuracy, or solidity of the contracts. In no event will CFGNINJA or its partners, employees, agents, or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions or actions with regards to the information provided in this audit report.

The assessment services provided by CFGNINJA are subject to dependencies and are under continuing development. You agree that your access 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. Cryptographic tokens are emergent technologies with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The services may access, and depend upon, multiple layers of third parties.