Security Assessment:

Grok Cat Token

April 11, 2024

- Audit Status: **Pass**
- Audit Edition: **Standard**

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.
ℹ️ Informational	Function Detected

Manual Code Review Risk Results

Contract Privilege	Description
🟢 Buy Tax	5%
🟢 Sale Tax	5%
🟢 Cannot Buy	Pass
🟢 Cannot Sale	Pass
🟡 Max Tax	20%
🟡 Modify Tax	Yes
🟡 Fee Check	Pass
🟢 Is Honeypot?	Not Detected
🟢 Trading Cooldown	Not Detected
🟢 Can Pause Trade?	Pass
🟢 Pause Transfer?	Not Detected
🟢 Max Tx?	Pass
🟢 Is Anti Whale?	Not Detected
🟡 Is Anti Bot?	Detected

Contract Privilege	Description
🟢 Is Blacklist?	Not Detected
🟢 Blacklist Check	Pass
🟡 is Whitelist?	Detected
🟢 Can Mint?	Pass
🟢 Is Proxy?	Not Detected
🟢 Can Take Ownership?	Not Detected
🟢 Hidden Owner?	Not Detected
ℹ️ Owner	0x93f705D7061Ff5A1Ed63AAc00BC4B972e80F725b
🟢 Self Destruct?	Not Detected
🟢 External Call?	Not Detected
🟢 Other?	Not Detected
🟢 Holders	0
🟠 Auditor Confidence	Medium-High Risk
🟢 KYC Present	Yes
🟢 KYC URL	https://www.assuredefi.com/projects/grok-cat/

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	0x4615C8E2db74517a34712C9BdEA5C418D999014B
Name	Grok Cat
Token Tracker	Grok Cat (GrokCat)
Decimals	9
Supply	4,200,000,000
Platform	BNBCHAIN
compiler	v0.8.19+commit.7dd6d404
Contract Name	GrokCat
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://bscscan.com/token/0x4615C8E2db74517a34712C9BdEA5C418D999014B#code
Payment Tx	Corporate

Main Contract Assessed
Contract Name

Name	Contract	Live
Grok Cat	0x4615C8E2db74517a34712C9BdEA5C418D999014B	Yes

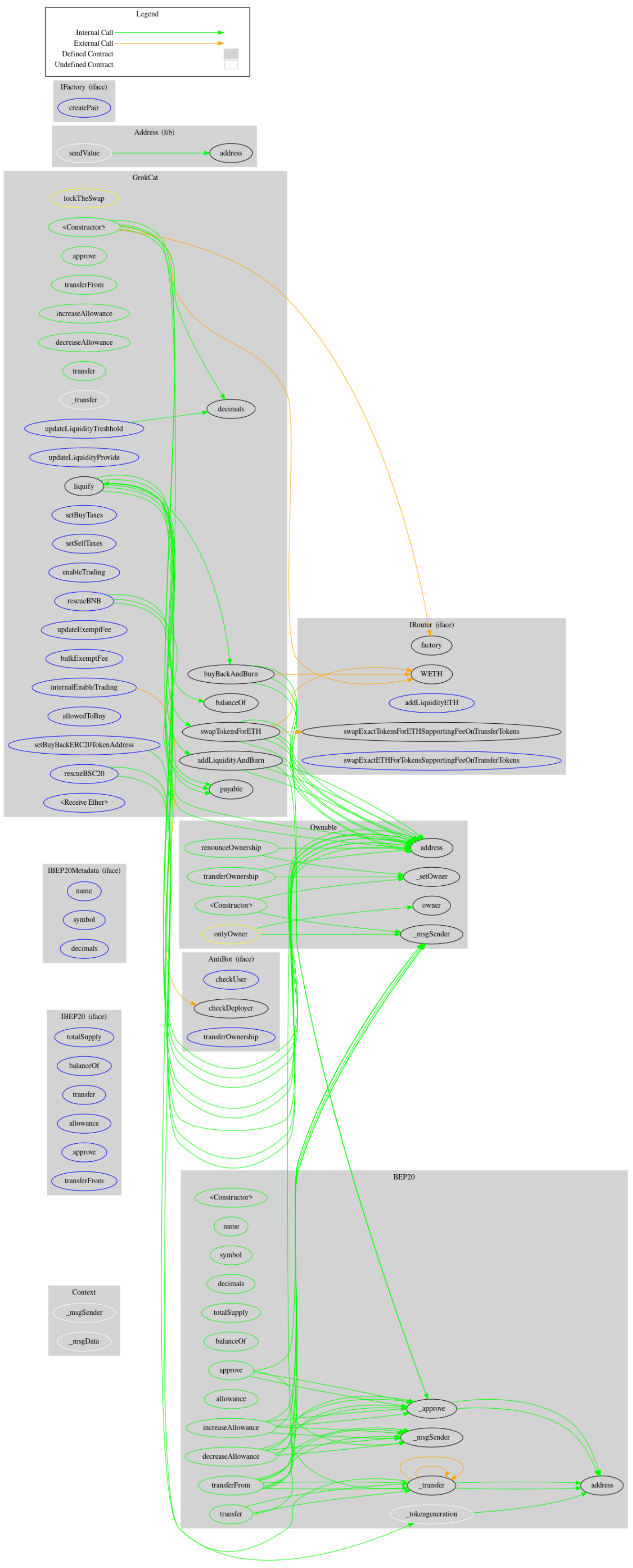
TestNet Contract was Not Assessed

Solidity Code Provided

SolID	File Sha-1	FileName
Grokcat	e975a3bef0da5a789080d7d1fa20ac8d98247fce	grokcat.sol
Grokcat		
Grokcat		
Grokcat		
Grokcat		
Grokcat	undefined	

Call Graph

The contract for Grok Cat has the following call graph structure.



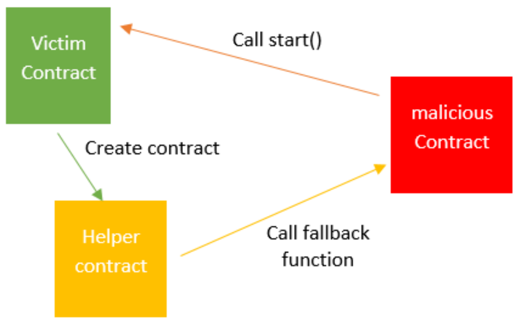
Reentrancy Check

The Project Owners of Grok Cat have not configure the Reentrancy Guard library.

You can read more about Reentrancy issues in the following link.
Reentrancy After Istanbul.

We recommend the team to add the library to the contract to avoid potential issues.

We recommend the team to create a new contract with Reentrancy Guard added to the same.



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	grokcat.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	grokcat.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	grokcat.sol	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	grokcat.sol	L: 16 C: 2
SWC-104	Pass	Unchecked Call Return Value.	grokcat.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	grokcat.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	grokcat.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	grokcat.sol	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	grokcat.sol	L: 0 C: 0
SWC-109	Pass	Uninitialized Storage Pointer.	grokcat.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	grokcat.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	grokcat.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	grokcat.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	grokcat.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	grokcat.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-115	Pass	Authorization through tx.origin.	grokcat.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	grokcat.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	grokcat.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	grokcat.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	grokcat.sol	L: 0 C: 0
SWC-120	Low	Potential use of block.number as source of randonmness.	grokcat.sol	L: 522 C: 40, L: 690-698 C: 6
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	grokcat.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	grokcat.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	grokcat.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	grokcat.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	grokcat.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	grokcat.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	grokcat.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	grokcat.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	grokcat.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U +202E).	grokcat.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	grokcat.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	grokcat.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	grokcat.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	grokcat.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	grokcat.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	grokcat.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-103 - Floating Pragma.

CWE-664: Improper Control of a Resource Through its Lifetime.

References:

Description:

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Remediation:

Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile locally.

References:

Ethereum Smart Contract Best Practices - Lock pragmas to specific compiler version.

Smart Contract Vulnerability Details

SWC-120 - Weak Sources of Randomness from Chain Attributes

CWE-330: Use of Insufficiently Random Values

Description:

Solidity allows for ambiguous naming of state variables when inheritance is used. Contract A with a variable x could inherit contract B that also has a state variable x defined. This would result in two separate versions of x, one of them being accessed from contract A and the other one from contract B. In more complex contract systems this condition could go unnoticed and subsequently lead to security issues.

Shadowing state variables can also occur within a single contract when there are multiple definitions on the contract and function level.

Remediation:

Using commitment scheme, e.g. RANDAO. Using external sources of randomness via oracles, e.g. Oraclize. Note that this approach requires trusting in oracle, thus it may be reasonable to use multiple oracles. Using Bitcoin block hashes, as they are more expensive to mine.

References:

How can I securely generate a random number in my smart contract?)

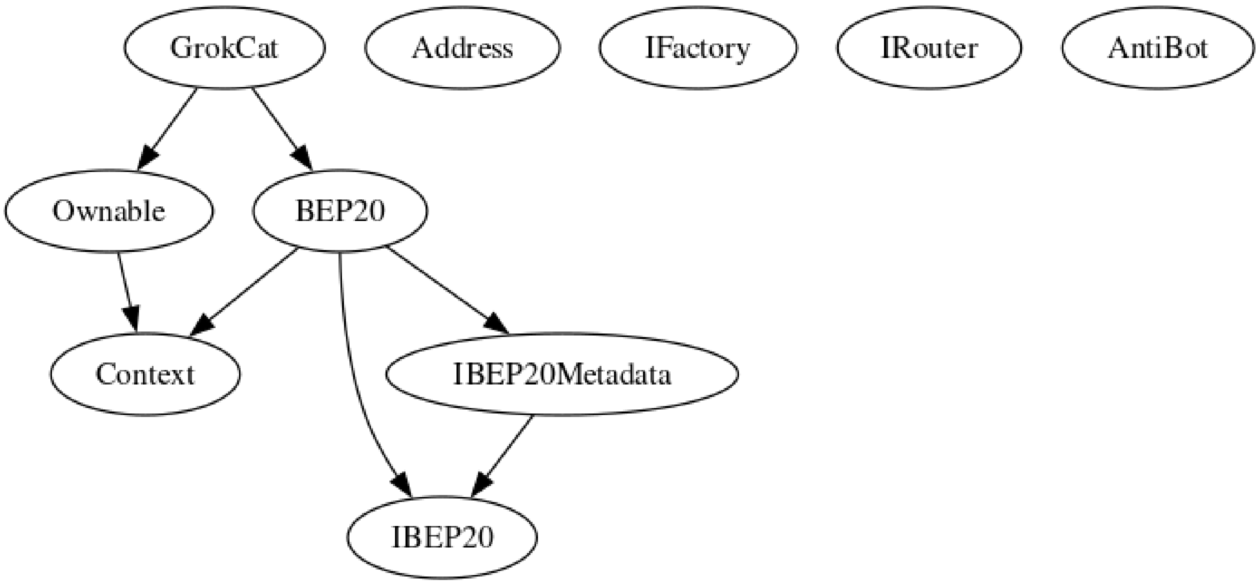
When can BLOCKHASH be safely used for a random number? When would it be unsafe?

The Run smart contract.

Inheritance

The contract for Grok Cat has the following inheritance structure.

The Project has a Total Supply of 4,200,000,000



Privileged Functions (onlyOwner)

Please Note if the contract is Renounced none of this functions can be executed.

Function Name	Parameters	Visibility
renounceOwnership		Public
transferOwnership	address newOwner	Public
updateLiquidityProvide	bool state	External
updateLiquidityTreshho ld	uint256 new_amount	External
setBuyTaxes	uint256 _lpBurn, uint256 _buyBackAndBurn, uint256 _walletSplit	External
setSellTaxes	uint256 _lpBurn, uint256 _buyBackAndBurn, uint256 _walletSplit	External
enableTrading		External
updateExemptFee	address _address, bool state	External
bulkExemptFee	address[] memory accounts, bool state	External
setBuyBackERC20Tok enAddress	address newBuyBack ERC20Token	External
allowedToBuy	address _address, bool state	External

Function Name	Parameters	Visibility
rescueBNB		External
rescueBSC20	address tokenAdd, uint256 amount	External

GrokCat-01 | Potential Sandwich Attacks.

Category	Severity	Location	Status
Security	🟡 Medium	grokcat.sol: L: 614, C: 14	Detected

Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction being attacked) a transaction to purchase one of the assets and make profits by back running (after the transaction being attacked) a transaction to sell the asset. The following functions are called without setting restrictions on slippage or minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the input amount is large:

- swapExactTokensForETHSupportingFeeOnTransferTokens()
- addLiquidityETH()

Remediation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

Referrences:

What Are Sandwich Attacks in DeFi — and How Can You Avoid Them?.

GrokCat-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	🟢 Low	grokcat.sol: L: 665-669 C: 14, L: 698-713 C:	Detected

Description

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

The given input is missing the check for the missing required function.

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. missing required function.

GrokCat-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	🟢 Low	grokcat.sol: L: 665-713 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.

GrokCat-08 | Dead Code Elimination.

Category	Severity	Location	Status
Coding Style	🟢 Low	grokcat.sol: L: 21 C:14, L: 313, L: 419 C: 14	Detected

Description

Functions that are not used in the contract, and make the code s size bigger.

```
_msgData()
ZERO
library Address
```

Remediation

Remove unused functions. dead-code elimination (also known as DCE, dead-code removal, dead-code stripping, or dead-code strip) is a compiler optimization to remove code which does not affect the program results. Removing such code has several benefits: it shrinks program size, an important consideration in some contexts, and it allows the running program to avoid executing irrelevant operations, which reduces its running time. It can also enable further optimizations by simplifying program structure.

https://docs.soliditylang.org/en/latest/cheatsheet.html

GrokCat-11 | Use of call for ETH Transfers.

Category	Severity	Location	Status
Security	🟢 Low	grokcat.sol: L: 3600-601 C: 14	Detected

Description

The Address.sendValue function uses a low-level call, which can be risky..

Remediation

Replace with transfer or send and ensure proper reentrancy guards.

Project Action

GrokCat-18 | Stop Transactions by using Enable Trade.

Category	Severity	Location	Status
Logical Issue	🔴 Critical	grokcat.sol: L: 684 C: 14	Detected

Description

Enable Trade is presend on the following contract and when combined with Exclude from fees it can be considered a whitelist process, this will allow anyone to trade before others and can represent and issue for the holders.

Remediation

We recommend the project owner to carefully review this function and avoid problems when performing both actions.

Project Action

GrokCat-19 | Centralization Risk via AntiBot.

Category	Severity	Location	Status
Trust & Cen tralization	🟡 Medium	grokcat.sol: L: 445 C: 14	Detected

Description

The internalEnableTrading function relies on an external AntiBot contract, introducing a single point of failure.

Remediation

Decentralize or provide transparency on the AntiBot contract's logic and ownership.

Project Action

GrokCat-20 | Arbitrary Token Buyback and Burn .

Category	Severity	Location	Status
Logical	🔴 Critical	grokcat.sol: L: 607 C: 14	Detected

Description

The buyBackAndBurn function could be exploited with a malicious buyBackERC20Token.

Remediation

Restrict the tokens that can be set for buyback or add validation checks.

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.
ℹ️ 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	2	2	0
🟠 High	0	0	1
🟡 Medium	2	2	0
🟢 Low	4	4	0
ℹ️ Informational	0	0	0
Total	8	9	0

Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/GrokCat_bsc	Pass
Other	no	N/A
Website	https://www.babygrok.ai/	Pass
Telegram	https://t.me/babygrok	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:

ASSURE DEFI™ | OFFICIAL PARTNER

Assessment Results

Score Results

Review	Score
Overall Score	85/100
Auditor Score	85/100
Review by Section	**Score**
Manual Scan Score	47
SWC Scan Score	33
Advance Check Score	5

The Following Score System Has been Added to this page to help understand the value of the audit, the maximum 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 84 Points for a higher standard, if a project does not attain 85% is an automatic failure. Read our notes and final assessment below.

Audit Passed

VERIFIED BY ASSURE DEFI ™ INTEGRITY ★ TRUST ★ CREDIBILITY

Assessment Results

Important Notes:

• Reentrancy Risk: Potential for attackers to withdraw funds repeatedly.ı

• External Contract Dependency: Reliance on outside contracts may introduce vulnerabilities.ı

• Centralization Risk: Significant control is held by the owner, creating a single point of failure.ı

• Use of call for ETH Transfers: Low-level operations increase the risk of security breaches.ı

• Arbitrary Token Buyback and Burn: The contract could be manipulated to buy back malicious tokens.ı

• Ownership Privileges: Extensive owner permissions could be dangerous if misused or compromised.ı

• Front-running Vulnerability: Transactions could be susceptible to exploitation by observant attackers.ı

• Launch Tax Logic: The initial tax setup could be exploited or lead to unexpected results.ı

• Fee Exemption Management: Certain addresses are exempt from fees, which could be unfairly applied.ı

• Hardcoded Addresses: Inflexibility due to preset addresses in the contract.ı

• Lack of Input Validation: Insufficient checks on inputs could lead to errors or attacks.ı

- No Time Lock on Sensitive Functions: Immediate execution of critical functions without delays.ı
- Conclusion: The contract exhibits several security and design concerns that need to be addressed. Centralization risks, reliance on external contracts, and potential for manipulation in fee and liquidity mechanisms are notable. The contract's use of hardcoded addresses and owner-centric controls also pose risks. While the contract has mechanisms to prevent reentrancy and provides basic token functionalities, improvements in security practices and reducing trust in centralized components are recommended. Implementing additional input validation, considering upgradeability, and ensuring transparent management of fee exemptions would enhance trust and security. It is advisable to conduct a comprehensive audit and testing before mainnet deployment to ensure the contract operates securely and as intended.

Auditor Score =85
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

Assure Defi 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 Assure Defi is under no covenant to audited completeness, accuracy, or solidity of the contracts. In no event will Assure Defi 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 Assure Defi 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.

ASSURE DEFI™
THE VERIFICATION **GOLD STANDARD**