CFG NINJA AUDITS

Security Assessment

BOJACK Token

May 9, 2023

Audit Status: Pass

Audit Edition: Advance

POWERED BY BLADE POOL

Table of Contents

Assessment Summary

This report has been prepared for BOJACK Token on the Binance Smart Chain network. CFGNINJA provides both client-centered and user-centered examination of the smart contracts and their current status when applicable. This report represents the security assessment made to find issues and vulnerabilities on the source code along with the current liquidity and token holder statistics of the protocol.

A comprehensive examination has been performed, utilizing Cross Referencing, Static Analysis, In-House Security Tools, and line-by-line Manual Review.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Inspecting liquidity and holders statistics to inform the current status to both users and client when applicable.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Verifying contract functions that allow trusted and/or untrusted actors to mint, lock, pause, and transfer assets.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders
- Thorough line-by-line manual review of the entire codebase by industry experts.

Project Overview

Token Summary

Parameter	Result
Address	0x339Fd7674cB25803E31Be976ec76cBBbF0e9f205
Name	BOJACK
Token Tracker	BOJACK (BOJACK)
Decimals	18
Supply	1,234,567,890
Platform	Binance Smart Chain
compiler	v0.8.17+commit.8df45f5f
Contract Name	Token
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://bscscan.com/tx/0xf08d3962160d5c676c80bd2aadae3c316b6524b97062dfd2d2244295ea463fef
Payment Tx	Corporate

Project Overview

Risk Analysis Summary

Parameter	Result
Buy Tax	0%
Sale Tax	0%
Is honeypot?	Clean
Trading Cooldown	No
Transfer Pausable	No
Modify Fees	Yes
Is anti whale?	No
Is blacklisted?	No
Is whitelisted?	No
Holders	0
Confidence Level	Low

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

Simulation Summary

Parameter	Result
Transfer From Owner	Pass
Transfer From Holder	Pass
Add Liquidity	Pass
Buy from Owner	Pass
Buy from Holder	Pass
Remove Liquidity	Pass
SwapAndLiquify	Pass
RemoveLiquidity	Pass
LaunchPad	PinkSale

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
BOJACK	0x339Fd7674cB25803E31Be976ec76cBBbF0e9f205	Yes

TestNet Contract Assessed Contract Name

Name	Contract	Live
BOJACK	0x1052f62741676Bf416804aD02080Da73EC08e1ef	Yes

Solidity Code Provided

SolID	File Sha-1	FileName
BoJack	71eefad84d016d2a73fc47dc784d81e0e70af0f5	Bojack.sol

Mint Check

The project owners of BOJACK do not have a mint function in the contract, owner cannot mint tokens after initial deploy.

The Project has a Total Supply of 1,234,567,890 and cannot mint any more than the Max Supply.

Mint Notes:

Auditor Notes:

Project Owner Notes:

Fees Check

The project owners of BOJACK do not have the ability to set fees higher than undefined .

The team May have fees defined; however, they can't set those fees higher than undefined or may not be able to configure the same.

Tax Fee Notes:

Auditor Notes: The contract currently has 0% buy and 0% sale taxes, and it cannot be set higher than 20%

Project Owner Notes:

Blacklist Check

The project owners of BOJACK do not have a blacklist function their contract.

The Project allow owners to transfer their tokens without any restrictions.

Token owner cannot blacklist the contract: Malicious or compromised owners can trap contracts relying on tokens with a blacklist.

Blacklist Notes:

Auditor Notes:

Project Owner Notes: undefined

MaxTx Check

The Project Owners of BOJACK cannot set max tx amount

The Team allows any investors to swap, transfer or sell their total amount if needed.

MaxTX Notes:

Auditor Notes: Customer has a max Wallet Configuration now.

Project Owner Notes:

Project Has No MaxTX

Pause Trade Check

The Project Owners of BOJACK don't have the ability to stop or pause trading.

The Team has done a great job to avoid stop trading, and investors has the ability to trade at any given time without any problems

Pause Trade Notes:

Auditor Notes:

Project Owner Notes:

Owner can't pause trading

Contract Ownership

The contract ownership of BOJACK is not currently renounced. The ownership of the contract grants special powers to the protocol creators, making them the sole addresses that can call sensible ownable functions that may alter the state of the protocol.

The current owner is the address 0xe8222921a07b931986045fd880af4c3032a3a937 which can be viewed:
HERE

The owner wallet has the power to call the functions displayed on the privileged functions chart below, if the owner's wallet is compromised, they could exploit these privileges.

We recommend the team renounce ownership at the right time, if possible, or gradually migrate to a timelock with governing functionalities regarding transparency and safety considerations.

We recommend the team use a Multisignature Wallet if the contract is not going to be renounced; this will give the team more control over the contract.

Liquidity Ownership

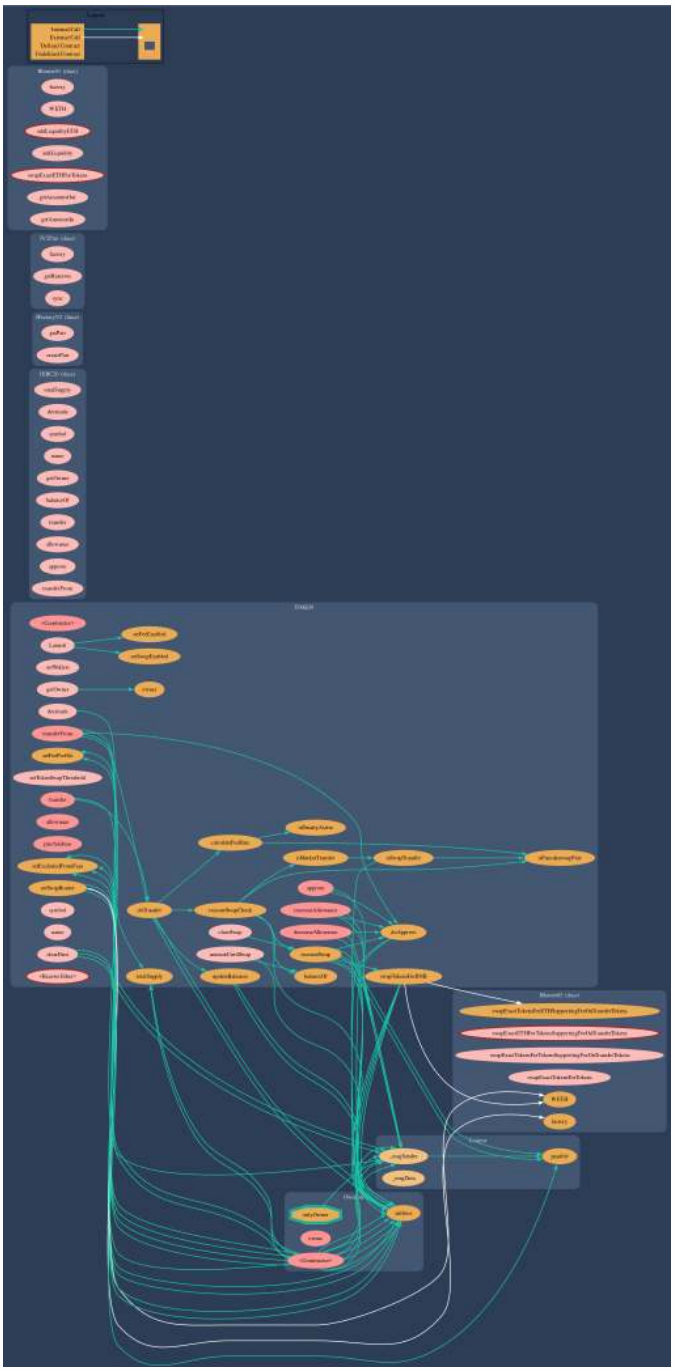
The token does not have liquidity at the moment of the audit, block 28061507

If liquidity is unlocked, then the token developers can do what is infamously known as 'rugpull'. Once investors start buying token from the exchange, the liquidity pool will accumulate more and more coins of established value (e.g., ETH or BNB or Tether). This is because investors are basically sending these tokens of value to the exchange, to get the new token. Developers can withdraw this liquidity from the exchange, cash in all the value and run off with it. Liquidity is locked by renouncing the ownership of liquidity pool (LP) tokens for a fixed time period, by sending them to a time-lock smart contract. Without ownership of LP tokens, developers cannot get liquidity pool funds back. This provides confidence to the investors that the token developers will not run away with the liquidity money. It is now a standard practice that all token developers follow, and this is what really differentiates a scam coin from a real one.

Read More

Call Graph

The contract for BOJACK has the following call graph structure.



KYC Information

The Project Owners of BOJACK is not KYC.

KYC Information Notes:

Auditor Notes:

Project Owner Notes:

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	Bojack.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	Bojack.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	Bojack.sol	L: 0 C: 0
SWC-103	Pass	A floating pragma is set.	Bojack.sol	L: 0 C: 0
SWC-104	Pass	Unchecked Call Return Value.	Bojack.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	Bojack.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	Bojack.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	Bojack.sol	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	Bojack.sol	L: 189 C: 9
SWC-109	Pass	Uninitialized Storage Pointer.	Bojack.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	Bojack.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-111	Pass	Use of Deprecated Solidity Functions.	Bojack.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	Bojack.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	Bojack.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	Bojack.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	Bojack.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	Bojack.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	Bojack.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	Bojack.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	Bojack.sol	L: 0 C: 0
SWC-120	Low	Potential use of block.number as source of randonmness.	Bojack.sol	L: 189 C: 25, L: 397 C: 89
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	Bojack.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	Bojack.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	Bojack.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	Bojack.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	Bojack.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-126	Pass	Insufficient Gas Griefing.	Bojack.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	Bojack.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	Bojack.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	Bojack.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U +202E).	Bojack.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	Bojack.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	Bojack.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	Bojack.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	Bojack.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	Bojack.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	Bojack.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-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:

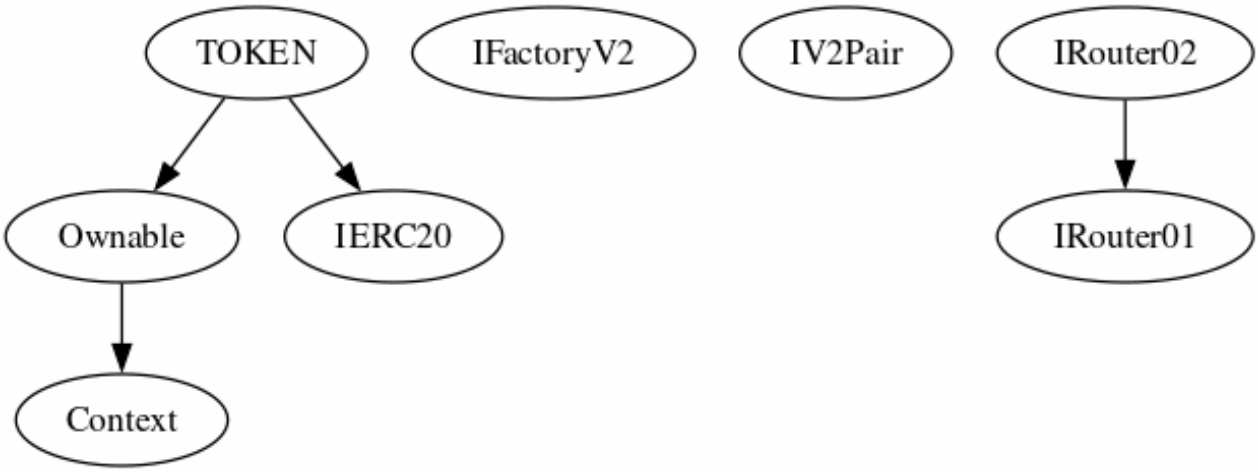
How can I securely generate a random number in my smart contract?)

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

The Run smart contract.

Inheritance

The contract for BOJACK has the following inheritance structure.



Privileged Functions (onlyOwner)

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

Function Name	Parameters	Visibility
renounceOwnership		public
transferOwnership	newOwner (address)	public
Launch	uint256 _blockPenalty	external
setNoFeeWallet	address account, bool enabled	external
setWallets	address _marketing, address _dev	external
setFeeProfile	uint8 _liquidity, uint8 _marketing, uint8 _dev, bool _updateBuy	public
clearSwap	uint256 _tokenAmount	external
setSwapRouter	address routerAddress	public

Function Name	Parameters	Visibility
setTokenSwapThreshold	uint256 threshold	Public
clearDust		public
setSwapEnabled	bool isEnabled	public
setFeeEnabled	bool isEnabled	public

Smart Contract Advance Checks

ID	Severity	Name	Result	Status
BOJACK-01	Minor	Potential Sandwich Attacks.	Fail	Resolved
BOJACK-02	Minor	Function Visibility Optimization	Pass	Resolved
BOJACK-03	Minor	Lack of Input Validation.	Fail	Pending
BOJACK-04	Major	Centralized Risk In addLiquidity.	Fail	Pending
BOJACK-05	Major	Missing Event Emission.	Pass	Resolved
BOJACK-06	Minor	Conformance with Solidity Naming Conventions.	Pass	Not-Found
BOJACK-07	Minor	State Variables could be Declared Constant.	Pass	Not-Found
BOJACK-08	Major	Dead Code Elimination.	Pass	Not-Found
BOJACK-09	Major	Third Party Dependencies.	Pass	Not Found
BOJACK-10	Major	Initial Token Distribution.	Pass	Resolved
BOJACK-11	Critical	The use of setHoldTime can lead to a pause trade or honeyPot State	Pass	Not-found
BOJACK-12	Major	Centralization Risks In The X Role	Pass	Not Found
BOJACK-13	Informational	Extra Gas Cost For User..	Fail	Not-Found
BOJACK-14	Medium	Unnecessary Use Of SafeMath	Pass	Not-Found

ID	Severity	Name	Result	Status
BOJACK-15	Medium	Symbol Length Limitation due to Solidity Naming Standards.	Pass	Not-Found
BOJACK-16	Medium	Invalid collection of Taxes during Transfer.	Pass	Not-Found

BOJACK-01 | Potential Sandwich Attacks.

Category	Severity	Location	Status
Security	Minor	Bojack.sol: 333, 14	Resolved

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?.

BOJACK-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	Minor	Bojack.sol: 395,14	Pending

Description

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

The given input is missing the check for the setFeeEnabled,clearSwap almost all functions is 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. setFeeEnabled,clearSwap almost all functions is missing required function.

BOJACK-04 | Centralized Risk In addLiquidity.

Category	Severity	Location	Status
Coding Style	Major	Bojack.sol: 286,13	Pending

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 BOJACK-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

liquidity is going try V2Router.addLiquidityETH{value: bnbToBeAddedToLiquidity}(address(this), tokensToAddAsLiquidity, 0, 0, _devWallet,

BOJACK-13 | Extra Gas Cost For User.

Category	Severity	Location	Status
Logical Issue	Informational	Bojack.sol: 249, 8	Not-Found

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

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.
Major	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
Minor	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.
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	0
Major	1	0	0
Medium	1	0	0
Minor	1	0	0
Informational	1	0	0
Total	4	0	5

Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/BojackH_BSC	Pass
Other		Fail
Website	https://bojackhorseman.io/	Pass
Telegram	https://t.me/BojackHorsemanBSC	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	86/100
Auditor Score	80/100
Review by Section	**Score**
Manual Scan Score	35/53
SWC Scan Score	36 /37
Advance Check Score	15 /19

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:

- No Vulnerabilities or issues were found.
- Several Coding practices are broken in the current code.
- Always DYOR.

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.