CFG NINJA AUDITS

Security Assessment

President PEPE Contract

May 4, 2023

Audit Status: Pass

Audit Edition: Advance

POWERED BY BLADE POOL

Table of Contents

Assessment Summary

This report has been prepared for President PEPE Contract 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	0x57eF27273ECA2Df2e0a6a0534e12bbAa26dd315d
Name	President PEPE
Token Tracker	President PEPE (PREPE)
Decimals	18
Supply	420,690,000,000,000
Platform	Binance Smart Chain
compiler	v0.8.19+commit.7dd6d404
Contract Name	Prepe
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://etherscan.io/address/0x57ef27273eca2df2e0a6a0534e12bbaa26dd315d#code
Payment Tx	0xe5466da7a152cddafbb167846e58a08565de579dceafb2eab3b651f1de094881

Project Overview

Risk Analysis Summary

Parameter	Result
Buy Tax	0%
Sale Tax	10%
Is honeypot?	Clean
Trading Cooldown	No
Transfer Pausable	Yes
Modify Fees	No
Is anti whale?	Yes
Is blacklisted?	Yes
Is whitelisted?	Yes
Holders	175
Confidence Level	High

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
President PEPE	0x57eF27273ECA2Df2e0a6a0534e12bbAa26dd315d	Yes

TestNet Contract Assessed Contract Name

Name	Contract	Live
President PEPE	0x4DB0a75897d53d36caB913aC6E7F6b9C0fB197C1	Yes

Solidity Code Provided

SolID	File Sha-1	FileName
prepe	caed812bbd73e65733abbe9d25bab5bfc1b1d9ff	prepe.sol

Mint Check

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

The Project has a Total Supply of 420,690,000,000,000 and cannot mint any more than the Max Supply.

Mint Notes:

Auditor Notes:

Project Owner Notes:

Fees Check

The project owners of President PEPE do not have the ability to set fees higher than 100 .

The team May have fees defined; however, they can't set those fees higher than 100 or may not be able to configure the same.

Tax Fee Notes:

Auditor Notes: The contract currently has 5% buy and 5% sale taxes, and cannot be set higher than 100% buy and 100% sale.

Project Owner Notes: Contract was renounced, no longer can set taxes.

Blacklist Check

The project owners of President PEPE 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: Contract is renounced, no longer can blacklist.

Project Owner Notes:

MaxTx Check

The Project Owners of President PEPE cannot set max tx amount

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

MaxTX Notes:

Auditor Notes: limits removed and contract renounced

Project Owner Notes:

Project Has No MaxTX

Pause Trade Check

The Project Owners of President PEPE 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: Contract renounced

Project Owner Notes: .

Owner can't pause trading

Contract Ownership

The contract ownership of President PEPE has been renounced or secured by a multi signature wallet. A multi signature wallet is one of the most secure method for a project.

Having no owner means that all the ownable functions in the contract can not be called by anyone, this often leads to more trust on the project.

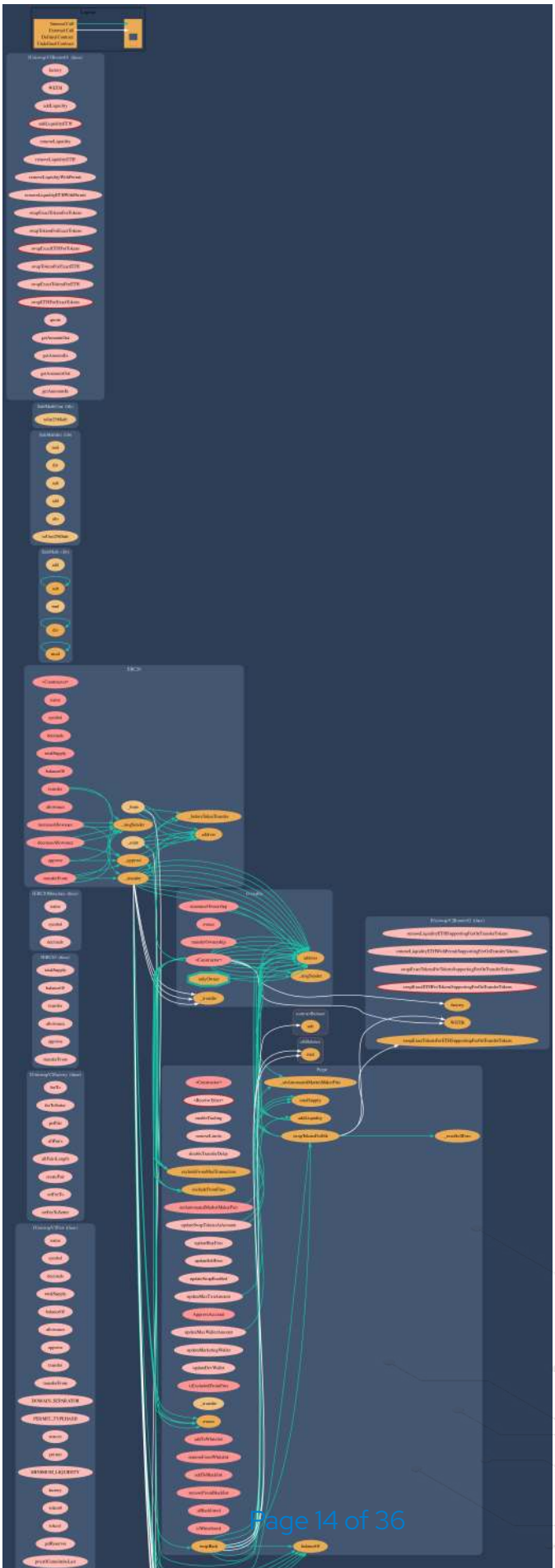
Liquidity Ownership

Most of the liquidity is currently locked; the lock can be seen here:

Liquidity Locker Link can be viewed from:
HERE

Call Graph

The contract for President PEPE has the following call graph structure.



KYC Information

The Project Owners of President PEPE is not KYC.

KYC Information Notes:

Auditor Notes: KYC to be completed by PinkSale, project will be a SAFU Project.

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	prepe.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	prepe.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	prepe.sol	L: 0 C: 0
SWC-103	Pass	A floating pragma is set.	prepe.sol	L: 0 C: 0
SWC-104	Pass	Unchecked Call Return Value.	prepe.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	prepe.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	prepe.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	prepe.sol	L: 0 C: 0
SWC-108	Low	State variable visibility is not set..	prepe.sol	L: 915 C: 9
SWC-109	Pass	Uninitialized Storage Pointer.	prepe.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	prepe.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-111	Pass	Use of Deprecated Solidity Functions.	prepe.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	prepe.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	prepe.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	prepe.sol	L: 0 C: 0
SWC-115	Low	Authorization through tx.origin.	prepe.sol	L: 1141 C: 38
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	prepe.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	prepe.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	prepe.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	prepe.sol	L: 0 C: 0
SWC-120	Low	Potential use of block.number as source of randonmness.	prepe.sol	L: 1010 C: 14
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	prepe.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	prepe.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	prepe.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	prepe.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	prepe.sol	L: 0 C: 0

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

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:

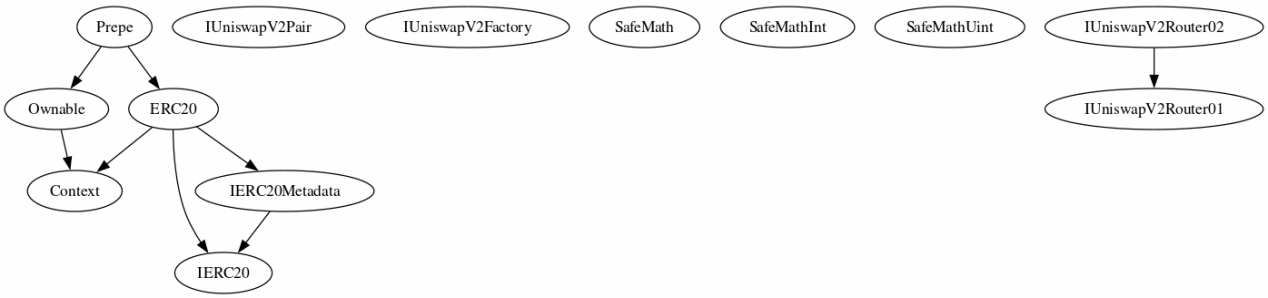
How can I securely generate a random number in my smart contract?)

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

The Run smart contract.

Inheritance

The contract for President PEPE 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	address newOwner	Public
enableTrading		External
removeLimits		External
disableTransferDelay		External
updateSwapTokensAtAmount		External
updateMaxTxnAmount		External
updateMaxWalletAmount		External
excludeFromMaxTransaction		External
updateBuyFees		External
updateSellFees		External

Function Name	Parameters	Visibility
updateSwapEnabled		External
excludeFromFees		Public
ApproveAccount		Public
setAutomatedMarketMakerPair		Public
updateMarketingWallet		External
updateDevWallet		External
addToWhitelist		External
removeFromWhitelist		External
addToBlacklist		External
removeFromBlacklist		External

Smart Contract Advance Checks

ID	Severity	Name	Result	Status
PREPE-01	Minor	Potential Sandwich Attacks.	Pass	Not-Found
PREPE-02	Minor	Function Visibility Optimization	Fail	Acknowledged
PREPE-03	Minor	Lack of Input Validation.	Fail	Acknowledged
PREPE-04	Major	Centralized Risk In addLiquidity.	Pass	Not-Found
PREPE-05	Minor	Missing Event Emission.	Fail	Acknowledged
PREPE-06	Minor	Conformance with Solidity Naming Conventions.	Pass	Not-Found
PREPE-07	Minor	State Variables could be Declared Constant.	Pass	Not-Found
PREPE-08	Minor	Dead Code Elimination.	Pass	Not-Found
PREPE-09	Major	Third Party Dependencies.	Pass	Not-Found
PREPE-10	Major	Initial Token Distribution.	Pass	Not-Found
PREPE-11	Major	dAPP Approval is set to all NFTs on the wallet an not limited to an specific contract.	Pass	Not-Found
PREPE-12	Major	Centralization Risks In The X Role	Pass	Not-Found
PREPE-13	Informational	Extra Gas Cost For User..	Pass	Not-Found
PREPE-14	Medium	Unnecessary Use Of SafeMath	Fail	Acknowledged

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

PREPE-02 | Function Visibility Optimization.

Category	Severity	Location	Status
Gas Optimization	Minor	prepe.sol: 408, 14	Acknowledged

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
launchedAt		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.

PREPE-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	Minor	prepe.sol: 1103,11	Acknowledged

Description

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

The given input is missing the check for the updateDevWallet,updateMarketingWallet,ApproveAccount,excludeFromFees, updateSwapEnabled, excludeFromMaxTransaction 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. updateDevWallet,updateMarketingWallet,ApproveAccount,excludeFromFees, updateSwapEnabled, excludeFromMaxTransaction is missing required function.

PREPE-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	Minor	prepe.sol: 613, 14	Acknowledged

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.

PREPE-14 | Unnecessary Use Of SafeMath

Category	Severity	Location	Status
Logical Issue	Medium	prepe.sol: 464,9	Acknowledged

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.
_balances[recipient] = _balances[recipient].add(amount);
magnifiedDividendPerShare = magnifiedDividendPerShare.add(
(amount).mul(magnitude) / totalSupply()
);
Note: Only a sample of 2 SafeMath library usage in this contract (out of 14) are shown above.

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.
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	0	0	0
Minor	2	0	
Informational	1	0	0
Total	4	0	0

Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/_presidentpepe	Pass
Other		Fail
Website	http://President-pepe.com	Pass
Telegram	https://t.me/presidentpepe	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	95/100
Auditor Score	80/100
Review by Section	**Score**
Manual Scan Score	53/53
SWC Scan Score	34 /37
Advance Check Score	8 /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 issues or vulnerabilities were found.
- The contract was tested and is fully functional.
- We will supervise the launch.
- Team will renounce and after will be a pass.

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.