



CFG NINJA AUDITS

Security Assessment

Pepe Tadpole Token

July 3, 2023

Audit Status: Pass

Audit Edition: Pinksale














POWERED BY
BLADE POOL

Risk Analysis

Classifications of Manual Risk Results

Classification	Description
 Critical	Danger or Potential Problems.
 Major	Be Careful or Fail test.
 Minor	Pass, Not-Detected or Safe Item.
 Informational	Function Detected

Manual Code Review Risk Results

Contract Priviledge	Description
 Buy Tax	4%
 Sale Tax	4%
 Cannot Sale	Pass
 Cannot Sale	Pass
 Max Tax	25%
 Modify Tax	Yes
 Fee Check	Pass
 Is Honeypot?	Not Detected.
 Trading Cooldown	Not Detected
 Can Pause Trade?	Pass
 Pause Transfer?	Not Detected



Contract Priviledge	Description
🟢 Max Tx?	Pass
🟢 Is Anti Whale?	Not 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
🔑 Owner	0xA11c6Af495A28515600F034A4241c260ab81d6B4
🟢 Self Destruct?	Not Detected
🟢 Other?	Not Detected
🟢 Other?	Not Detected
🟢 Holders	1
🟢 Auditor Confidence	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

Token Summary

Parameter	Result
Address	0xA993e7cF7C48e6Bf3A735B9D10e91baAF812eab9
Name	Pepe Tadpole
Token Tracker	Pepe Tadpole (PEPETAD)
Decimals	9
Supply	100,000,000,000,000
Platform	Binance Smart Chain
compiler	v0.8.4+commit.c7e474f2
Contract Name	LiquidityGeneratorToken
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://etherscan.io/address/0x624a6bec7f1362dc3c06c517a6ab54fc98b0229e#code
Payment Tx	0xd5b68153e6c3cd0b74c16a35d2c493064a683ce5dfdc40085f8ab92ed85b59bb



Main Contract Assessed

Contract Name

Name	Contract	Live
Pepe Tadpole	0xA993e7cF7C48e6Bf3A735B9D10e91baAF812eab9	Yes

TestNet Contract was Not Assessed

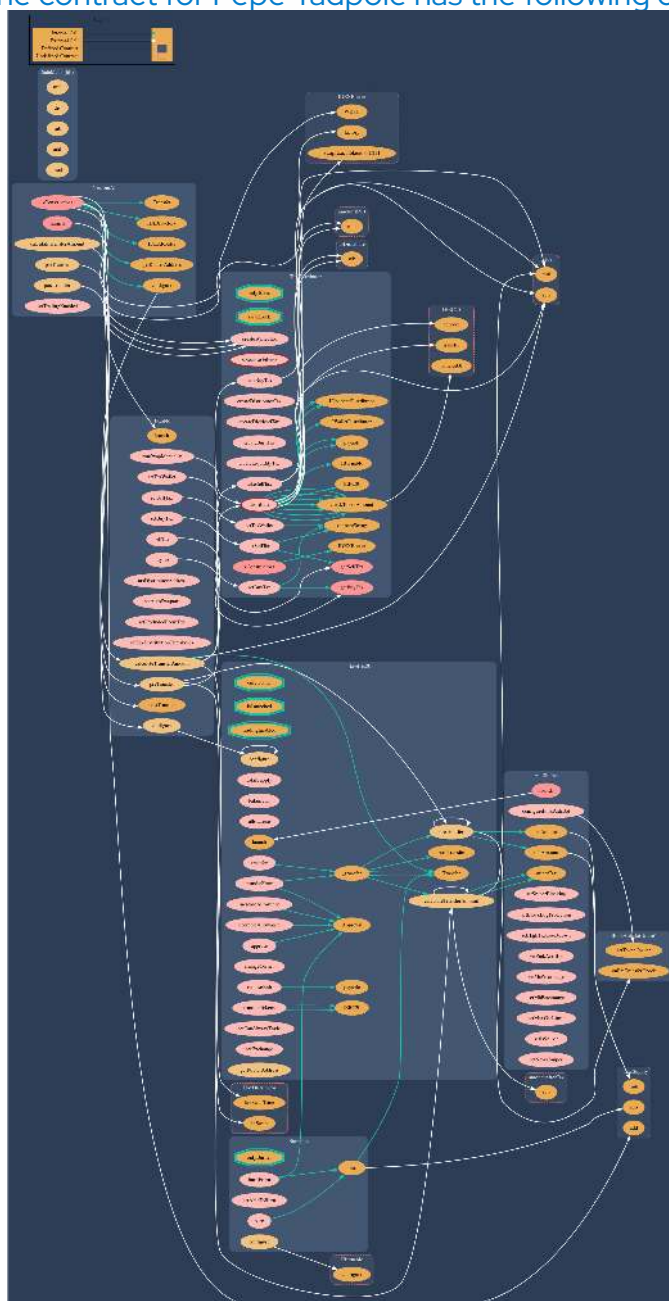
Solidity Code Provided

SolidID	File Sha-1	FileName
PEPETAD	2e65744db35a6703eb5e2b7817ae7f642d527b44	PEPETAD.sol
PEPETAD		
PEPETAD		
PEPETAD		



Call Graph

The contract for Pepe Tadpole 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	LiquidityGenerator.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	LiquidityGenerator.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	LiquidityGenerator.sol	L: 0 C: 0
SWC-103	Pass	A floating pragma is set.	LiquidityGenerator.sol	L: 0 C: 0
SWC-104	Pass	Unchecked Call Return Value.	LiquidityGenerator.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	LiquidityGenerator.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	LiquidityGenerator.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	LiquidityGenerator.sol	L: 0 C: 0
SWC-108	Low	State variable visibility is not set..	LiquidityGenerator.sol	L: 959 C: 9
SWC-109	Pass	Uninitialized Storage Pointer.	LiquidityGenerator.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	LiquidityGenerator.sol	L: 0 C: 0



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



ID	Severity	Name	File	location
SWC-125	Pass	Incorrect Inheritance Order.	LiquidityGenerator.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	LiquidityGenerator.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	LiquidityGenerator.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	LiquidityGenerator.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	LiquidityGenerator.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U+202E).	LiquidityGenerator.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	LiquidityGenerator.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	LiquidityGenerator.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	LiquidityGenerator.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	LiquidityGenerator.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	LiquidityGenerator.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	LiquidityGenerator.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.

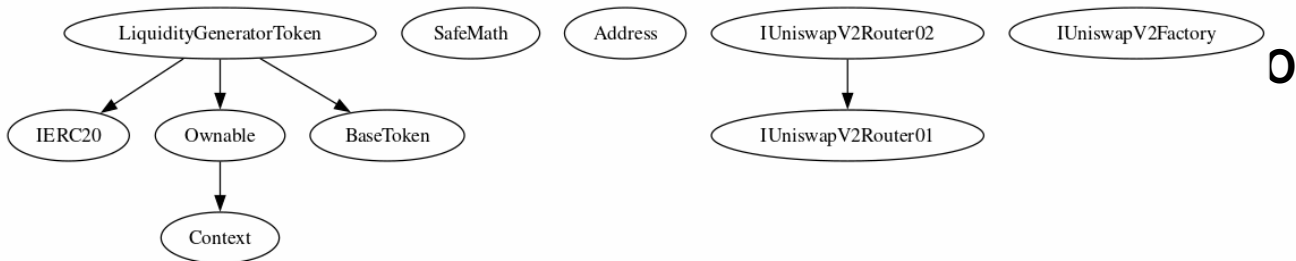
References:

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



Inheritance

The contract for Pepe Tadpole has the following inheritance structure.



Smart Contract Advance Checks



ID	Severity	Name	Result	Status
PEPETAD-01	Low	Potential Sandwich Attacks.	Pass	Not-Found
PEPETAD-02	Informational	Function Visibility Optimization	Pass	Detected
PEPETAD-03	Low	Lack of Input Validation.	Pass	Detected
PEPETAD-04	High	Centralized Risk In addLiquidity.	Pass	Not-Found
PEPETAD-05	Low	Missing Event Emission.	Pass	Detected
PEPETAD-06	Low	Conformance with Solidity Naming Conventions.	Pass	Not-Found
PEPETAD-07	Low	State Variables could be Declared Constant.	Pass	Not-Found
PEPETAD-08	Low	Dead Code Elimination.	Pass	Not-Found
PEPETAD-09	High	Third Party Dependencies.	Pass	Detected
PEPETAD-10	High	Initial Token Distribution.	Pass	Not-Found
PEPETAD-11	High	claimStuckTokens can claim own tokens.	Pass	Detected
PEPETAD-12	High	Centralization Risks In The X Role	Pass	Not-Found
PEPETAD-13	Informational	Extra Gas Cost For User..	Pass	Detected



ID	Severity	Name	Result	Status
PEPETAD-6	Informational	Unnecessary Use Of SafeMath	Fail	Pending
PEPETAD-15	Medium	Symbol Length Limitation due to Solidity Naming Standards.	Pass	Detected
PEPETAD-16	Medium	Invalid collection of Taxes during Transfer.	Pass	Not Detected
PEPETAD-17	Logical Issue	Conformance to numeric notation best practice.	Pass	Detected
PEPETAD-18	Critical	Stop Transactions by using Enable Trade.	Pass	Not Detected



PEPETAD-14 | Unnecessary Use Of SafeMath

Category	Severity	Location	Status
Logical Issue	 Informational	LiquidityGenerator.sol: 205, 10	 Pending

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.
 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	NaN	undefined	undefined
 Medium	0	0	0
 Minor	NaN	undefined	undefined
 Informational	0	0	0
Total	NaN	0	0



Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/pepetadpole?t=MqTVoby6ofwT9ACG170_A&s=09	Pass
Other		Fail
Website	https://pepetadpole.biz	Pass
Telegram	https://t.me/pepetadpole	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:



Audit Result

Final Audit Score

Review	Score
Security Score	85
Auditor Score	80

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 must 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.
- This is a Pinksale Liquidity Generator token, please review the tax structure.
- 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 act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM 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 how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being 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 may result in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the 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 requirements on the input variables than a setter function.

Coding Best Practices

ERC 20 Coding Standards are a set of rules that each developer should follow to ensure the code meets a set of criteria 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 advocacy 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.

