Revoluzion Audit

Audit Report

Name	: Catboy
Symbol	: CATBOY
Decimals	: 18
Address	: 0x0DceE5F694E492F0DD842a7FBE5BEd4C6E4665a6
Owner	: 0xCE26459680fc272c5678D35b40F838291e68550f
Network	: Binance Smart Chain
Type	: ERC20
Audited on	: 14 February 2024
Audited Score	: 72%

Revoluzion Audit

Table of Contents

Revoluzion Audit

Project Overview

Name	Catboy
Symbol	CATBOY
Decimals	18
Total Supply	100,000,000
Tax	6% For Buy & Sell
Compiler Version	v0.8.15+commit.e14f2714
Optimization	Yes with 200 runs
License Type	UNLICENSED
Explorer Link	https://bscscan.com/address/0x0dcee5f694e492f0dd842a7fbe5bed4c6e4665a6
Create Tx	https://bscscan.com/tx/0x7bcd1bd069e890250a732ee2b65367cb9104d4f5901daaec95353765eadf1b47
Creator	0xCE26459680fc272c5678D35b40F838291e68550f
Featured Wallet	AutoLiquidityReceiver = 0x76F16e60C2822468e1193775Fc8F10a0bfa3a9a9; MarketingFeeReceiver = 0xbd8f6ae2a3a18eE63A32d831f68D3C5dfA58F39C; OpsFeeReceiver = 0x7Aeb56E110d8bD60Bd9a1d24FBa82af8C0D63d7c; StakingFeeReceiver = 0x0e7e43f1F9D8cD05c5BeF2c37b52B774A2B2B697;

	DevFeeReceiver = 0x5A3454A2a6BAcdB4316041875500f7eF6B83E873; SecDevFeeReceiver = 0xDd9D9D02e8C90FcD122959b7e3579Bc588594B83;

Project Description

According to their website

No information provided.

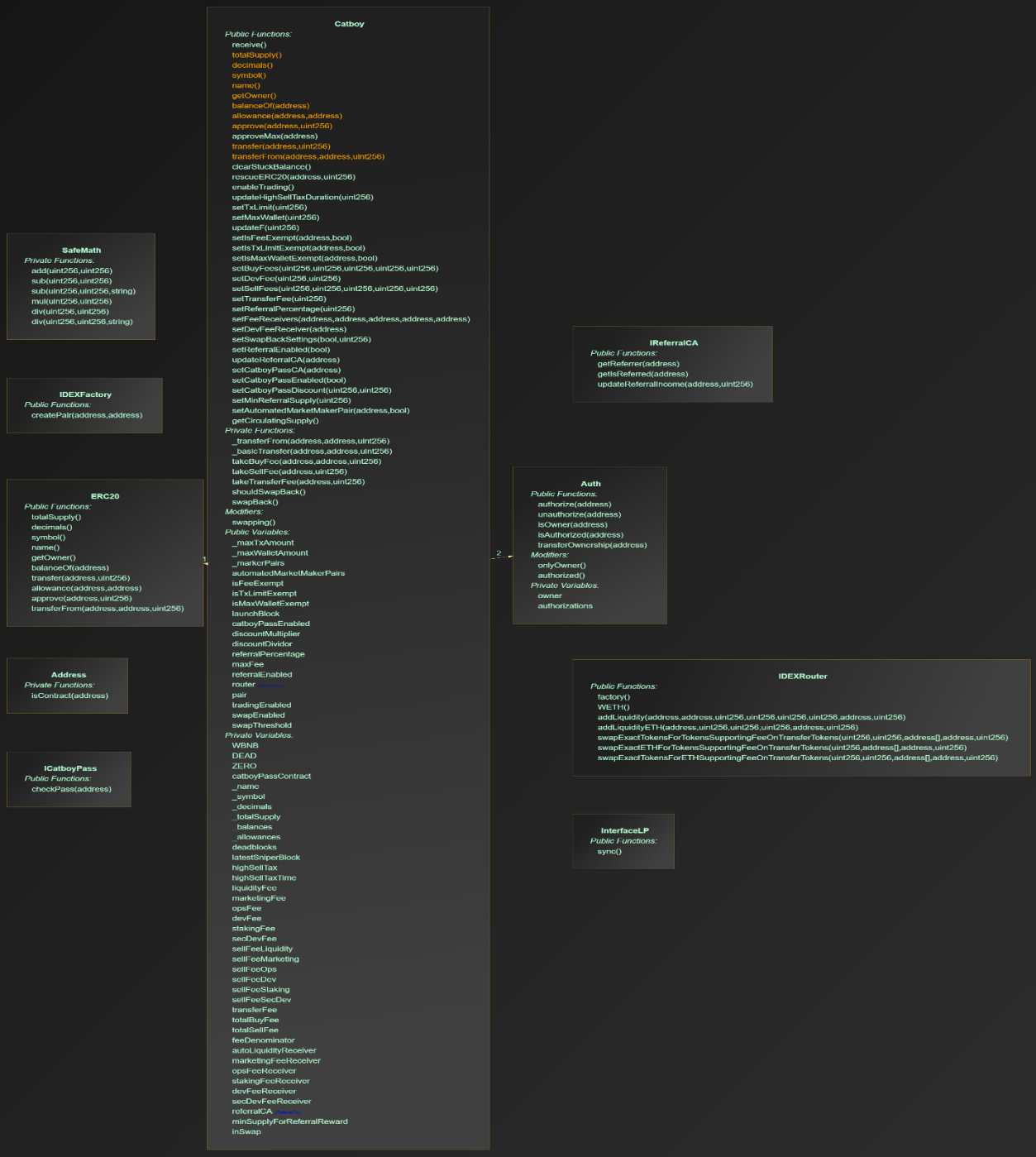
Release Date : TBA

Category : DeFi

Revoluzion Audit

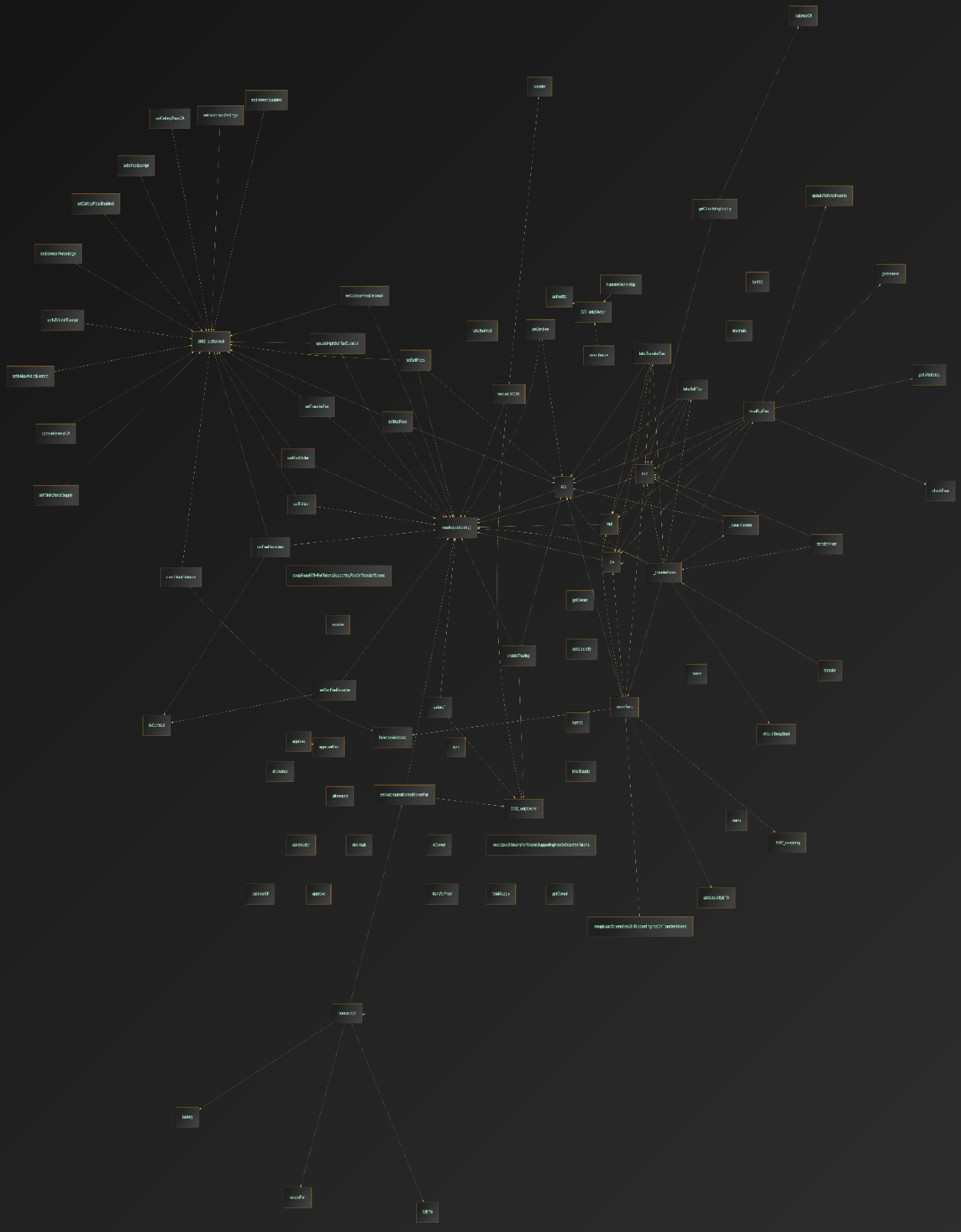
Contract Functions Interaction

Inheritance Graph



Revoluzion Audit

Call Graph (All)



Audit Overview

Threat Level

When conducting audit on smart contract(s), we first look for known vulnerabilities and issues within the code because any exploitation on such vulnerabilities and issues by malicious actors could potentially result in serious financial damage to the projects. All the issues and vulnerabilities will be categorized into the categories as provided below.

Critical

This category provides issues and vulnerabilities that are critical to the performance/functionality of the smart contract and should be fixed by project creator before moving to a live environment.

Medium

This category provides issues and vulnerabilities that are not that critical to the performance/functionality of the smart contract but is recommended to be fixed by project creator before moving to a live environment.

Minor

This category provides issues and vulnerabilities that are minor to the performance/functionality of the smart contract and can remain unfixed by project creator before moving to a live environment.

Informational

This category provides issues and vulnerability that have insignificant effect on the performance/functionality of the smart contract and can remain unfixed by project creator before moving to a live environment. However, fixing them can further improve the efficacy or security for features with a risk-free factor.

Notable Information

- Project owner and users should be aware that the smart contract owner will be able to withdraw any token from the smart contract except the native token that was collected from transaction taxes.

- Project owner and users should be aware that no transaction can be made between two addresses that are not exempted from fee if the trade has yet to be enabled by the smart contract owner.

- Project owner and users should be aware that this smart contract implemented an anti sniper logic, thus please take note that the fee during the time before it reached highSellTaxTime will be a total of 20% for both buy and sell transaction instead of a total of 7% for buy and 6% for sell transaction as set upon deployment.

- Project owner and users should be aware that only the dev address will be able to initiate setDevFeeReceiver function with any address except zero address.

- Project owner and users should be aware that the owner of the smart contract will only be able to renounce the ownership of the smart contract by manually transferring the ownership to the zero address using transferOwnership.

- Project owner and users should be aware that the owner of the smart contract will be able to use any address, be it a wallet address or any other smart contract address, when initiating setAutomatedMarketMakerPair function since there are no restriction to ensure that only the pair smart contract address can be used when initiating the function.

- New smart contract owner should be aware that the previous owner address will remain as an authorized address unless it has manually been manually unauthorized since the transferOwnership function does not automatically do so.

- Smart contract owner need to remember not to use the current address when initiating transferOwnership function since the function to change the

address does not have any restriction to prevent such action and it will be a waste of gas for the initiator.

- Smart contract owner need to remember not to use the current value when initiating updateF function since the function to change the value does not have any restriction to prevent such action and it will be a waste of gas for the initiator.

- Smart contract owner need to remember not to initiate updateF function after the trade has already been enabled since the function does not have any restriction to prevent such action and it will be a waste of gas for the initiator as the updated value will no longer be used in any part of the smart contract after the trade was enabled.

- Dev account need to remember not to use the his/her current dev address when initiating setDevFeeReceiver function since the function to change the address does not have any restriction to prevent such action and it will be a waste of gas for the initiator.

- Dev account need to remember not to use all the current values when initiating setDevFee function since the function to change the values does not have any restriction to prevent such action and it will be a waste of gas for the initiator.

- Authorized accounts need to remember not to use the current address when initiating updateReferralCA and setCatboyPassCA function since the function to change the address does not have any restriction to prevent such action and it will be a waste of gas for the initiator.

- Authorized accounts need to remember not to use the current value when initiating updateHighSellTaxDuration, setTxLimit, setMaxWallet, setTransferFee, setReferralPercentage and setMinReferralSupply function since the function to change the value does not have any restriction to prevent such action and it will be a waste of gas for the initiator.

- Authorized accounts need to remember not to use the current state when initiating setReferralEnabled and setCatboyPassEnabled function since the function to change the state does not have any restriction to prevent such action and it will be a waste of gas for the initiator.

- Authorized accounts need to remember not to use the current state for an address when initiating setIsFeeExempt, setIsMaxWalletExempt and setIsTxLimitExempt function since the function to change the state does not have any restriction to prevent such action and it will be a waste of gas for the initiator.

- Authorized accounts need to remember not to use all the current value when initiating setBuyFees, setSellFees, setSwapBackSettings and setCatboyPassDiscount function since the function to change the values does not have any restriction to prevent such action and it will be a waste of gas for the initiator.

- Authorized accounts need to remember not to use all the current address when initiating setFeeReceivers function since the function to change the addresses does not have any restriction to prevent such action and it will be a waste of gas for the initiator.

- Authorized accounts need to remember not to initiate updateHighSellTaxDuration function after the trade has already been enabled since the function does not have any restriction to prevent such action and it will be a waste of gas for the initiator as the updated value will no longer be used in any part of the smart contract after the trade was enabled.

Caution Information

- Project owner and users should be aware that any address that has been added into the list of authorized addresses will be able to withdraw the ether balance available in the smart contract to their address since there is no restriction implemented in the clearStuckBalance to only allow the transfer of ether only to a specific address.

- Project owner and users should be aware that this audit does not include any vulnerabilities that could potentially occur in conjunction to the logic of the function from referralCA or catboyPassContract since the source code of the said contracts were not provided to be audited along with this smart contract.

- Project owner and users should be aware that if the balance of the native token in the smart contract is too high, the smart contract might not be able to swap the token due to the price impact and this could potentially lead to the smart contract to turn into a honeypot since the swap function will be taking the whole balance of the native token available in the smart contract.

- Users should be aware that the smart contract implemented the trading enable logic when only smart contract owner can initiate enableTrading function to allow the trade between two addresses that are not exempted from fee and no transaction can be made for this scenario if the smart contract did not enable the trade and thus resulting in this smart contract to be a honeypot.

Bugs and Optimizations Detection

This table is based on the result obtained from running the smart contract through Slither's Solidity static analysis.

What it detects	Impact	Confidence	Status
Storage abiencoderv2 array	High	High	Passed
transferFrom uses arbitrary from	High	High	Passed
Modifying storage array by value	High	High	Passed
The order of parameters in a shift instruction is incorrect.	High	High	Passed
Multiple constructor schemes	High	High	Passed
Contract's name reused	High	High	Passed
Detected unprotected variables	High	High	Passed
Public mappings with nested variables	High	High	Passed
Right-To-Left-Override control character is used	High	High	Passed
State variables shadowing	High	High	Passed
Functions allowing anyone to destruct the contract	High	High	Passed
Uninitialized state variables	High	High	Passed
Uninitialized storage variables	High	High	Passed

Revoluzion Audit

Unprotected upgradeable contract	High	High	Passed
transferFrom uses arbitrary from with permit	High	Medium	Passed
Functions that send Ether to arbitrary destinations	High	Medium	Moderated
Tainted array length assignment	High	Medium	Passed
Controlled delegatecall destination	High	Medium	Passed
Payable functions using delegatecall inside a loop	High	Medium	Passed
msg.value inside a loop	High	Medium	Passed
Reentrancy vulnerabilities (theft of ethers)	High	Medium	Moderated
Signed storage integer array compiler bug	High	Medium	Passed
Unchecked tokens transfer	High	Medium	Passed
Weak PRNG	High	Medium	Passed
Detects ERC20 tokens that have a function whose signature collides with EIP-2612's DOMAIN_SEPARATOR()	Medium	High	Passed
Detect dangerous enum conversion	Medium	High	Passed
Incorrect ERC20 interfaces	Medium	High	Passed
Incorrect ERC721 interfaces	Medium	High	Passed
Dangerous strict equalities	Medium	High	Passed

Revoluzion Audit

Contracts that lock ether	Medium	High	Passed
Deletion on mapping containing a structure	Medium	High	Passed
State variables shadowing from abstract contracts	Medium	High	Passed
Tautology or contradiction	Medium	High	Passed
Unused write	Medium	High	Moderated
Misuse of Boolean constant	Medium	Medium	Passed
Constant functions using assembly code	Medium	Medium	Passed
Constant functions changing the state	Medium	Medium	Passed
Imprecise arithmetic operations order	Medium	Medium	Passed
Reentrancy vulnerabilities (no theft of ethers)	Medium	Medium	Passed
Reused base constructor	Medium	Medium	Passed
Dangerous usage of tx.origin	Medium	Medium	Passed
Unchecked low-level calls	Medium	Medium	Passed
Unchecked send	Medium	Medium	Passed
Uninitialized local variables	Medium	Medium	Moderated
Unused return values	Medium	Medium	Moderated
Modifiers that can return the default value	Low	High	Passed

Revoluzion Audit

Built-in symbol shadowing	Low	High	Passed
Local variables shadowing	Low	High	Passed
Uninitialized function pointer calls in constructors	Low	High	Passed
Local variables used prior their declaration	Low	High	Passed
Constructor called not implemented	Low	High	Passed
Multiple calls in a loop	Low	Medium	Passed
Missing Events Access Control	Low	Medium	Passed
Missing Events Arithmetic	Low	Medium	Passed
Dangerous unary expressions	Low	Medium	Passed
Missing Zero Address Validation	Low	Medium	Moderated
Benign reentrancy vulnerabilities	Low	Medium	Passed
Reentrancy vulnerabilities leading to out-of-order Events	Low	Medium	Moderated
Dangerous usage of block.timestamp	Low	Medium	Moderated
Assembly usage	Informational	High	Moderated
Assert state change	Informational	High	Passed
Comparison to boolean constant	Informational	High	Moderated
Deprecated Solidity Standards	Informational	High	Passed
Un-indexed ERC20 event parameters	Informational	High	Passed

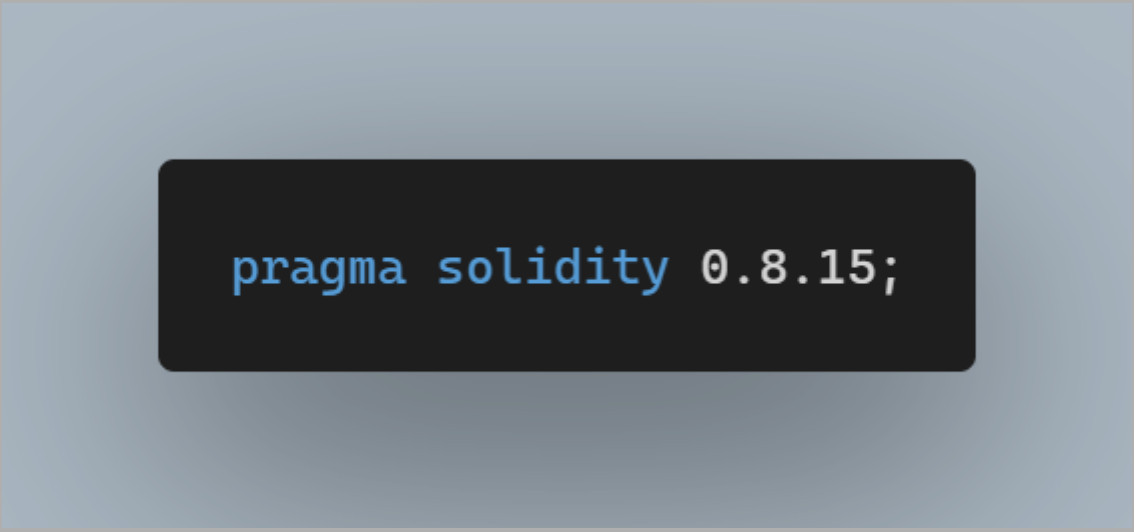
Revoluzion Audit

Function initializing state variables	Informational	High	Moderated
Low level calls	Informational	High	Moderated
Missing inheritance	Informational	High	Passed
Conformity to Solidity naming conventions	Informational	High	Moderated
If different pragma directives are used	Informational	High	Passed
Redundant statements	Informational	High	Passed
Incorrect Solidity version	Informational	High	Passed
Unimplemented functions	Informational	High	Passed
Unused state variables	Informational	High	Passed
Costly operations in a loop	Informational	Medium	Moderated
Functions that are not used	Informational	Medium	Passed
Reentrancy vulnerabilities through send and transfer	Informational	Medium	Passed
Variable names are too similar	Informational	Medium	Moderated
Conformance to numeric notation best practices	Informational	Medium	Moderated
State variables that could be declared constant	Optimization	High	Passed
Public function that could be declared external	Optimization	High	Moderated

Contract Diagnostic

CODE	SEVERITY	DESCRIPTION
ISV	Informational	Incorrect Solidity version

SV — Incorrect Solidity Version

SEVERITY	Informational
LOCATION(S)	L3
	
DESCRIPTION	We identified incorrect Solidity versions used in the contract, which can result in compatibility issues and potential vulnerabilities.
RECOMMENDATIONS	We recommend exercising caution when using incorrect Solidity versions in your contract, as they can result in compatibility issues and potential vulnerabilities. Ensure that your contract's Solidity version is appropriate and compatible with the desired functionality

Disclaimer

This report only shows findings based on our limited project analysis according to the good industry practice from the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, overall online presence and team transparency details of which are set out in this report. To get a full view of our analysis, **it is important for you to read the full report**. Under no circumstances did Revoluzion Audit receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. **Our team provides no guarantees against the sale of team tokens or the removal of liquidity by the project** audited in this document.

While **we have done our best to conduct thorough analysis to produce this report**, it is crucial to note that you should not rely solely on this report and use the content provided in this document as financial advice or a reason to buy any investment. The Our team disclaims any liability against us for the resulting losses based on the you decision made by relying on the content of this report. **You must conduct your own independent investigations before making any decisions** to protect yourselves from being scammed. We go into more detail on this in the disclaimer clause in the next page — please make sure to read it in full.

Full Disclaimer Clause

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy all copies of this report downloaded and/or printed by you. This report is provided for information purposes only, on a non-reliance basis and does not constitute to any investment advice.

No one shall have any right to rely on the report or its contents, and Revoluzion Audit and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (collectively known as Revoluzion) owe no duty of care towards you or any other person, nor do we make any warranty or representation to any person on the accuracy or completeness of the report.

The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Revoluzion hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report.

Except and only to the extent that it is prohibited by law, Revoluzion hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Revoluzion, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts, website, social media, and team.