Revoluzion Audit

Audit Report

Name	: Honk 2.0
Symbol	: HONK2.0
Decimals	: 18
Address	: 0xFb79F415Fc1dA28D99d757C02bdFe696d7833eD1
Owner	: 0x167aC1c1d38768208B39dDB4a0Ea196e4BC7c69C
Network	: Binance Smart Chain
Type	: ERC20
Audited on	: 30 January 2024
Audited Score	: 67%

Revoluzion Audit

Table of Contents

Project Overview

Name	Honk 2.0
Symbol	HONK2.0
Decimals	18
Total Supply	420,000,000,000,000
Tax	Buy Tax 1% Buy Tax | 1% To Marketing Wallet Sell Tax 2% Sell Tax | 2% To Marketing Wallet Transfer Tax 3% Transfer Tax | 3% To Marketing Wallet
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	Yes with 200 runs
License Type	MIT
Explorer Link	https://bscscan.com/address/0xFb79F415Fc1dA28D99d757C02bdFe696d7833eD1
Create Tx	https://bscscan.com/tx/0x36e0c047264781420330be7554ff43ef41bcbcd0afea3f4c33a1980c468843cd
Creator	0x167aC1c1d38768208B39dDB4a0Ea196e4BC7c69C
Featured Wallet	Marketing - 0x9551d3be196D7a756800323A5d58FE4Eb2468ceB

Project Description

According to their website

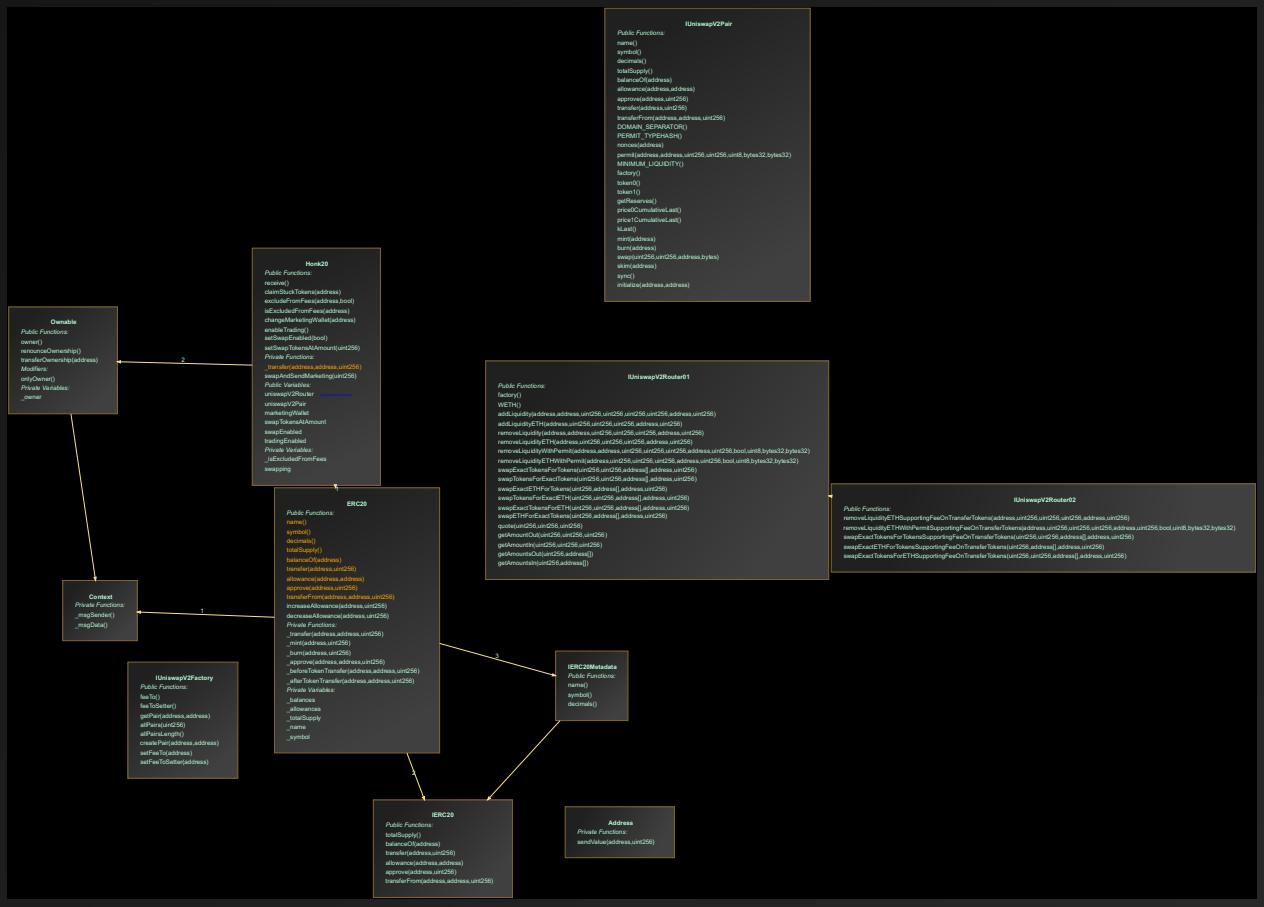
Honk 2.0, the meme god, bestows prosperty on it's followers with 5% taxes for all Honk 2.0 transactions. NoFounderTokens, NoTreasuryTokens, NoAdvisorsTokens, NoVCs. Rest will be using for PRESALE, BURNT and CEX.

Release Date : TBA

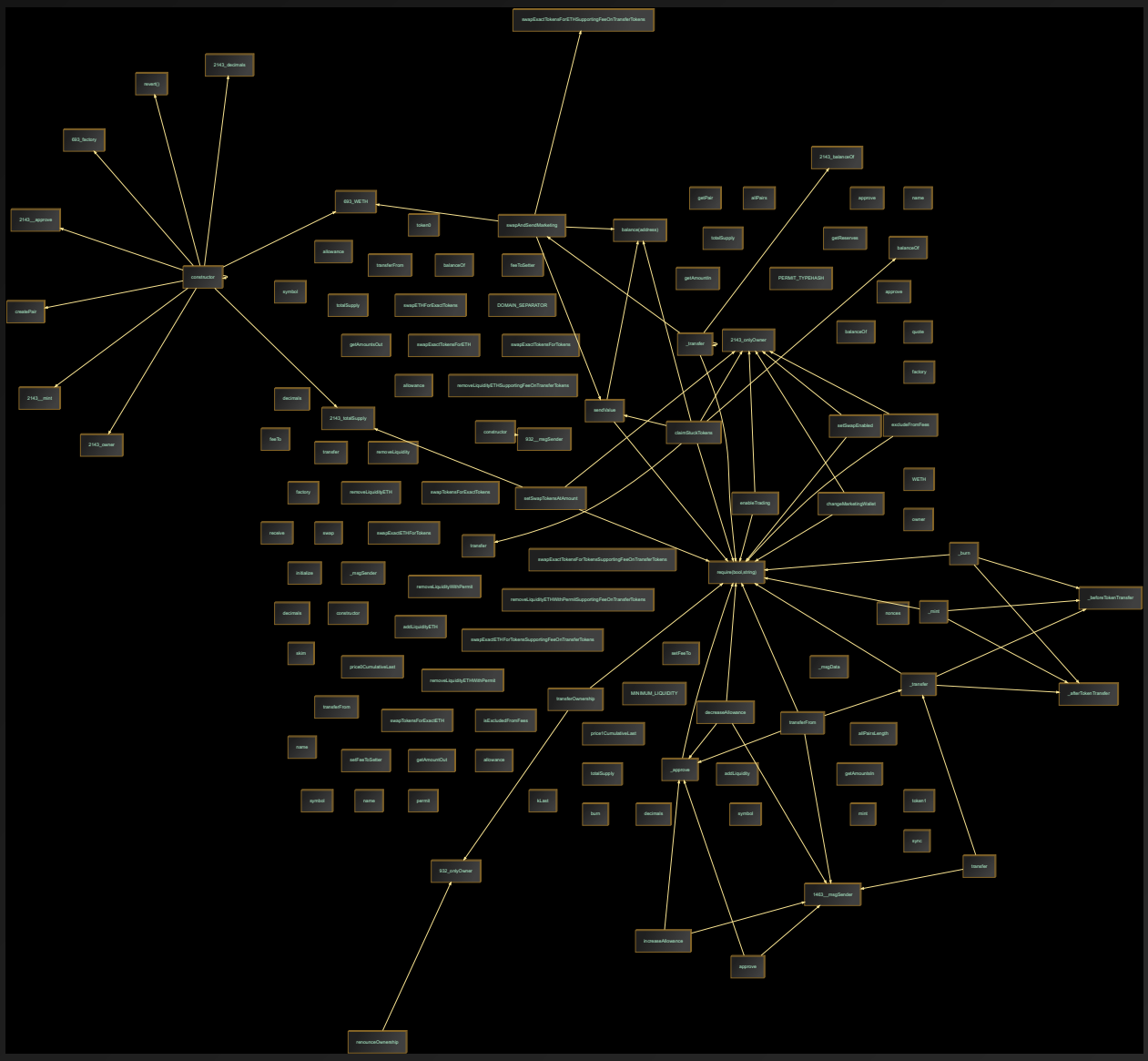
Category : DeFi

Contract Functions Interaction

Inheritance Graph



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

- Contract Owner cannot stop or pause transactions.
- Contract Owner cannot transfer tokens from specific address.
- Contract Owner cannot mint new tokens after deploying smart contract.
- Contract Owner cannot burn tokens from specific wallet.
- Contract Owner cannot blacklist wallet.
- There are no compiler warnings when compiling the smart contracts.
- Contract is using safe Zeppelin modules.
- Contract is a standard ERC20 token with buy, sell, transfer tax to marketing wallet. No max txn or max wallet.
- Project owner should be aware that this smart contract can only be used for deployment on Ethereum and Binance Smart Chain mainnet in addition to the Ethereum Goerli and Binance Smart Chain testnet due to the logic implemented in the costructor for contract deployment
- Project owner should be aware that if the price impact is too high, the smart contract won't be able to swap the token if the balance in the smart contract is too high, which will most likely to turn this smart contract into a honeypot from there on since there's no fail safe logic implemented to fix this potential situation.
- Project owner and users should be aware that any transaction involving zero address will not be possible be it for buy, sell or transfer since the restriction logic has been hardcoded in _transfer function.
- Project owner and users should be aware that the fee logic was hardcoded into the smart contract as 1% fee for buy transaction from pair address stored as uniswapV2Pair and 2% fee for sell transaction to

pair address stored as uniswapV2Pair in addition to the 3% fee for all other transactions involving two addresses that are not excluded from fee.

- Project owner and users should be aware that the claimStuckTokens will not be able to rescue any token that does not adhere to the ERC20 standard since it is not using the safeTransfer module from SafeERC20 when transferring the token.

- Smart contract owner need to remember not to set the current address for _marketingWallet when initiating changeMarketingWallet since the function to change the address does not have a restriction to prevent such action which is just a waste of gas for the owner.

- Smart contract owner need to remember not to set the current value for swapTokensAtAmount when initiating setSwapTokensAtAmount since the function to change the value does not have a restriction to prevent such action which is just a waste of gas for the owner.

Cautionary Information

- **Smart contract owner need to remember not to initiate changeMarketingWallet function with an address of a smart contract that cannot receive any ether funds as this will result in this token smart contract to turn into a honeypot since the swap function will failed and no transaction, whether it is transfer, buy or sell can actually go through although this would also mean that the project owner will also not receive any fund if this happened.**

- Users should be aware that the smart contract include enableTrading function that will prevent any kind of transaction involving two addresses that are not excluded from fees as long as the owner of the smart contract didn't enable the trade, which can only be triggered once only if the trade was not yet enabled. Can be ignored if its stealth launch or post launch.

- Project owner and users should be aware that for a buy or sell transaction that involved other pair addresses that could potentially exist such as HONK2.0/BUSD or HONK2.0/USDT as examples, there will be a flat fee of 3% incurred for both type of transactions since the smart contract will be using the fee logic for normal transfer since such addresses for the pair are not being stored in the smart contract to allow the smart contract to detect them for the application of the normal fee of 1% for buy transaction and 2% for sell transaction unless the said pair smart contract address is excluded from fee, which will then result in no fee incurred on any of the buy or sell transaction involving that pair address.

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

Unprotected upgradeable contract	High	High	Passed
transferFrom uses arbitrary from with permit	High	Medium	Passed
Functions that send Ether to arbitrary destinations	High	Medium	Passed
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	Moderated
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

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	Passed
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

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	Passed
Benign reentrancy vulnerabilities	Low	Medium	Passed
Reentrancy vulnerabilities leading to out-of-order Events	Low	Medium	Moderated
Dangerous usage of block.timestamp	Low	Medium	Passed
Assembly usage	Informational	High	Passed
Assert state change	Informational	High	Passed
Comparison to boolean constant	Informational	High	Passed
Deprecated Solidity Standards	Informational	High	Passed
Un-indexed ERC20 event parameters	Informational	High	Passed

Function initializing state variables	Informational	High	Passed
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	Moderated
Incorrect Solidity version	Informational	High	Moderated
Unimplemented functions	Informational	High	Passed
Unused state variables	Informational	High	Passed
Costly operations in a loop	Informational	Medium	Passed
Functions that are not used	Informational	Medium	Moderated
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	Passed
State variables that could be declared constant	Optimization	High	Passed
Public function that could be declared external	Optimization	High	Passed

Contract Diagnostic

CODE	SEVERITY	DESCRIPTION
SWC-110	Unknown	Out of bound array access.

SWC-110 — Out of bound array access

SEVERITY	Unknown
LOCATION(S)	L599-L618

```
function swapAndSendMarketing(uint256 tokenAmount) private {
        uint256 initialBalance = address(this).balance;

        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = uniswapV2Router.WETH();

        uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
            tokenAmount,
            0,
            path,
            address(this),
            block.timestamp);

        uint256 newBalance = address(this).balance - initialBalance;

        payable(marketingWallet).sendValue(newBalance);

        emit SwapAndSendMarketing(tokenAmount, newBalance);
    }
```

DESCRIPTION	Based on our analysis, the code tries to access an element of an array using an index that is outside its bounds.
RECOMMENDATIONS	We recommend to ensure that all array accesses are within the bounds of the array to avoid similar issues in the future. Should have no issues as this is quite the standard for solidity.

	However, there should be no issues with using such logic provided that the project owner did not use the index for the path array that exceed the array length.
STATUS	**Revoluzion acknowledgement:** It's important to note that using such logic is permissible, and there should be no issues as long as the project owner does not exceed the array's length when accessing elements. Your commitment to following these best practices is commendable, and we believe it will contribute to the project's stability and reliability.

Constructor Calls

```
constructor () ERC20("Honk 2.0", "HONK2.0")
   {
      address router;
      if (block.chainid == 56) {
          router = 0x10ED43C718714eb63d5aA57B78B54704E256024E; // BSC Pancake Mainnet Router
      } else if (block.chainid == 97) {
          router = 0xD99D1c33F9fC3444f8101754aBC46c52416550D1; // BSC Pancake Testnet Router
      } else if (block.chainid == 1 || block.chainid == 5) {
          router = 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D; // ETH Uniswap Mainnet % Testnet
      } else {
          revert();
     }

      IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(router);
      address _uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
          .createPair(address(this), _uniswapV2Router.WETH());

      uniswapV2Router = _uniswapV2Router;
      uniswapV2Pair   = _uniswapV2Pair;

      _approve(address(this), address(uniswapV2Router), type(uint256).max);

      marketingWallet = 0x9551d3be196D7a756800323A5d58FE4Eb2468ceB;

      _isExcludedFromFees[owner()] = true;
      _isExcludedFromFees[address(0xdead)] = true;
      _isExcludedFromFees[address(this)] = true;
      _isExcludedFromFees[0x407993575c91ce7643a4d4cCACc9A98c36eE1BBE] = true; //pinklock

      _mint(owner(), 420e12 * (10 ** decimals()));
      swapTokensAtAmount = totalSupply() / 5_000;

      tradingEnabled = false;
      swapEnabled = false;
 }
```

Summary

Revoluzion has conducted a comprehensive analysis of the smart contract deployed address: **0xF9afcbd52c19ceF59c6de6944977ECa95c3850Dd**, and we acknowledged that it exhibits reliable security and functionality. The smart contract has been **verified on Sepolia (ETH Testnet)**, which demonstrates its transparency and ensures the integrity of the underlying code.

Our analysis revealed that the **smart contract has implemented several security measures** to mitigate potential risks. It does not contain any unsafe modules, upgradeable or proxy functions, or any possible disruptive features to pause the smart contract. These design choices contribute to the overall security and stability of the contract.

During the smart contract audit, we have identified that **certain functionalities may require some adjustments to enhance security**. It is important to note that the smart contract includes cautionary functions, such as allowing the Address with the operator role to burn BidNow tokens from a specific address and mint BidNow tokens to a specific address. These functions have been implemented with careful consideration and adherence to security protocols.

It is important to note that **regular audits and ongoing vigilance are crucial for maintaining the safety and optimal performance** of the contract. We recommend conducting periodic security assessments to address any emerging threats and to ensure continuous protection. Should you have any further inquiries or require additional support, **please do not hesitate to contact us**. We are committed to provide an ongoing assistance and to ensure a long-term security and success for your smart contract.

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.