



# TOKEN GUARD

[more details: tokenguard.io](https://tokenguard.io)





# TOKEN GUARD

## Table of contents:

1. Tokenguard methodology.....	3
2. Contact background.....	4
3. Project summary.....	4
4. Project overview.....	5
5. List of security checks.....	5
6. Single Code Verification results.....	7
7. Liability clause.....	10
8. Contact details.....	11



# ABOUT TOKENGUARD

**TokenGuard** aggregates the best techniques for smart contracts automated verification. We are helping teams design a secure smart contract / token and monitor the source code for new vulnerabilities and prevent potential hacks and scams. TokenGuard provides a basic critical vulnerability verification in less than 12 hours. (We cover 29/36 of the most important security checks from SWC Registry).

Our mission is to **make blockchain safe**.

## Methodology:

We're using different tools for vulnerability discovery that are based on 3 technologies - **symbolic execution**, **static analysis** and **fuzzing**. All of them are specifically designed for Ethereum Virtual Machine (EVM), which is the smart contract execution environment. It doesn't matter whether the contract logic is focused on token creation or swapping - the tools are set to find critical vulnerabilities that may exist in each type of contract.

Our auditors verify the output of the tools and deliver information whether there is vulnerability in the code.

[more details: tokenguard.io](https://tokenguard.io)



# ANALYSIS REPORT

## Background:

Michał from CleanCarbon contacted us in a need to perform a smart contract source code verification for their project. CleanCarbon is a BEP20 token designated for combining physical energy-from-waste installations with the crypto world. The contract address is: **0xa52262dA176186105199a597aC8Cf094FF71b0C5**. We provided the code verification of their solidity contract using our Tokenguard engine and manual review. The team provided us with the .sol files.

## Project summary:

<b>Project:</b>	CleanCarbon
<b>Blockchain:</b>	BSC
<b>Language:</b>	Solidity
<b>Contract name</b>	CarboToken
<b>Compiler version:</b>	v0.8.0
<b>Website:</b>	<a href="https://cleancarbon.io">https://cleancarbon.io</a>
<b>Request date:</b>	29/03/2022
<b>Report date:</b>	06/04/2022
<b>Max supply:</b>	500 000 000



# ANALYSIS REPORT











## OVERVIEW:

CleanCarbon is a crypto-funded, community-driven crypto project that aims to clean our planet. It's a DeFi response to the world's pollution.

The project consists of two elements the CARBO token and the physical Waste-to-Energy installations

Both elements work together and support each other, bridging cryptocurrencies and the traditional physical installations.

## List of security checks:

<b>SWC - 100</b> Function Default Visibility	
<b>SWC - 101</b> Integer Overflow and Underflow	
<b>SWC - 102</b> Outdated Compiler Version	
<b>SWC - 103</b> Floating Pragma	
<b>SWC - 104</b> Unchecked Call Return Value	
<b>SWC - 105</b> Unprotected Ether Withdrawal	
<b>SWC - 106</b> Unprotected SELFDESTRUCT Instruction	
<b>SWC - 107</b> Reentrancy	
<b>SWC - 108</b> State Variable Default Visibility	
<b>SWC - 109</b> Uninitialized Storage Pointer	

[more details: tokenguard.io](https://tokenguard.io)



# ANALYSIS REPORT

**SWC - 110** Assert Violation



**SWC - 111** Use of Deprecated Solidity Functions



**SWC - 112** Delegatecall to Untrusted Callee



**SWC - 113** DoS with Failed Call



**SWC - 114** Transaction Order Dependence



**SWC - 115** Authorization through tx.origin



**SWC - 116** Block values as a proxy for time



**SWC - 117** Signature Malleability



**SWC - 119** Shadowing State Variables



**SWC - 120** Weak Sources of Randomness from Chain Attributes



**SWC - 124** Write to Arbitrary Storage Location



**SWC - 125** Incorrect Inheritance Order



**SWC - 127** Arbitrary Jump with Function Type Variable



**SWC - 128** DoS With Block Gas Limit



**SWC - 129** Typographical Error



**SWC - 130** Right-To-Left-Override control character (U+202E)



# ANALYSIS REPORT

SWC - 131 Presence of unused variables



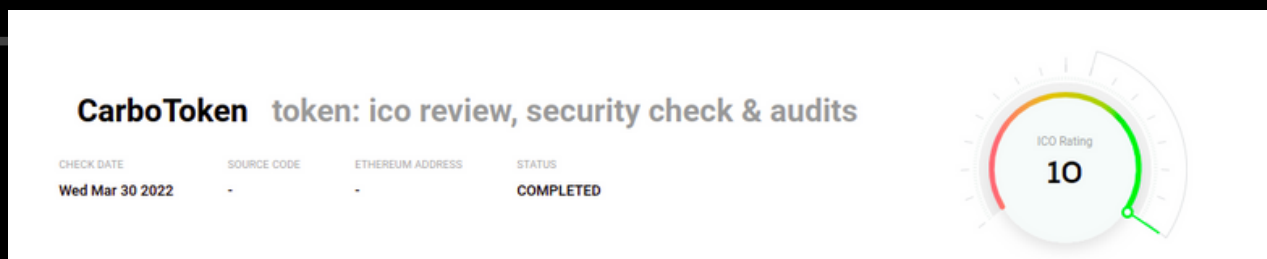
SWC - 132 Unexpected Ether balance



SWC - 135 Code With No Effects



## Summary:



The following report presents the effect of the Tokenguard basic source code vulnerabilities analysis. This security check performed on 05/04/2022 based on .sol files provided by CleanCarbon.

We have checked: CarboToken.sol, RecoverableFunds.sol, WithCallback.sol, VestingWallet.sol, FeeManager.sol, DividendManager.sol, CrowdSale.sol, Configurator.sol, FeeHolder.sol

The smart contracts were analyzed for basic critical smart contract vulnerabilities, exploits and manipulation hacks according to [swcregistry.io](https://www.swcregistry.io).

[more details: tokenguard.io](https://www.tokenguard.io)



# ANALYSIS REPORT

The CleanCarbon's basic critical vulnerability verification has found the following issues:

1. Reentrancy issue in DividendManager.sol file, function: distributeDividends.
2. Reentrancy issue in DividendManager.sol file, function: excludeFromDividends.
3. Reentrancy issue in VestingWallet.sol file, function: deposit (uint256 schedule, address[] calldata beneficiaries, uint256[] calldata amounts)
4. Reentrancy issue in VestingWallet.sol file, function: deposit (uint256 schedule, address beneficiary, uint256 amount)

Those remarks were shared with the CleanCarbon Tech Team and they responded:

*"All 4 comments relate to the same case: re-entry attack. The case will work only if the administrator purposefully replaces the address of the token with the address of another ERC-20 token, in which the transferFrom method contains code to attack our smart contract.*

*Accordingly, the vulnerability can be implemented only if the attacker received the owner's private key.*

[online version: here](#)

[more details: tokenguard.io](#)





# ANALYSIS REPORT

*CleanCarbon team has decided there is no risk in leaving the code without any changes."*

The project has passed the verification and is



Verified with  
TokenGuard

# TOKEN GUARD

[online version: here](#)

[more details: tokenguard.io](https://tokenguard.io)



# ABOUT TOKENGUARD

**Tokenguard.io** is the first automated rating agency for Ethereum and other blockchains. We use the most sophisticated tools to find bugs in tokens that would allow hackers take over your funds.

**Tokenguard.io** supports the construction of secure blockchain infrastructure for fintech and enterprise customers around the world.

## LIABILITY CLAUSE

Please note that Tokenguard.io doesn't verify the economic foundation of the project but only its code correctness and security issues. We do not take any responsibility for any misuse or misunderstanding of the information provided and potential economic losses due to faulty investment decisions. This document doesn't ensure that the code itself is free from potential vulnerabilities that were not found. If any questions arise please contact us via [www.tokenguard.io](http://www.tokenguard.io).





# TOKEN GUARD

**Thank you for your attention!**

in terms of any additional questions please contact: [tom@tokenguard.io](mailto:tom@tokenguard.io)

[more details: tokenguard.io](https://tokenguard.io)

