



AUDIT REPORT

SecureWise

TED CRYPTO
(CRYPTOTED)



<https://securewise.info/>



<https://t.me/securewisehub>



<https://github.com/securewise>

Contents

- 
- 02 Disclaimer
 - 03 Overview
 - 04 Quick Result
 - 05 Auditing Approach and Methodologies
 - 06 Automated Analysis
 - 08 Inheritance Graph
 - 10 Manual Review

Disclaimer

SecureWise provides the smart contract audit of solidity. Audit and report are for informational purposes only and not, nor should be considered, as an endorsement to engage with, invest in, participate, provide an incentive, or disapprove, criticise, discourage, or purport to provide an opinion on any particular project or team.

This audit report doesn't provide any warranty or guarantee regarding the nature of the technology analysed. These reports, in no way, provide investment advice, nor should be used as investment advice of any sort. Investors must always do their own research and manage their risk.

DISCLAIMER: 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 any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and SecureWise and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) SecureWise owe no duty of care towards you or any other person, nor does SecureWise 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 SecureWise 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, SecureWise hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against SecureWise, 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.

Overview

Token Name: CryptoTed (**CRYPTOTED**)

Methodology: Automated Analysis, Manual Code Review

Language: Solidity

Contract Address: 0xc2E04588eE034d2260bD24Ae74065152D212Ea42

ContractLink: <https://etherscan.io/address/0xc2E04588eE034d2260bD24Ae74065152D212Ea42>

Network: Ethereum (ETH)

Supply: 3000000000000

Website: <https://cryptoted.vip>

Twitter: <https://twitter.com/realtedcrypto>




Telegram: <https://t.me/therealtedcrypto>

Report Date: May 21, 2023

Quick Result



SecureWise has applied the automated and manual analysis of Smart Contract and were reviewed for common contract vulnerabilities and centralized exploits

	The owner can pause trading
	The owner can add/remove whitelisted address
	The owner can set pair address

Page 10 for more details

CryptoTed (CRYPTOTED) as **FAILED** the smart contract audit with **HIGH** and **LOW** severity issue

Auditing Approach and Methodologies

SecureWise has performed starting with analyzing the code, issues, code quality, and libraries. Reviewed line-by-line by our team. Finding any potential issue like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

Methodology

- Understanding the size, scope and functionality of your project's source code
- Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Testing and automated analysis of the Smart Contract to determine proper logic has been followed throughout the whole process
- Deploying the code on testnet using multiple live test
- Analyzing a program to determine the specific input that causes different parts of a program to execute its functions.
- Checking whether all the libraries used in the code are on the latest version.

Goals

Smart Contract System is secure, resilient and working according to the specifications and without any vulnerabilities.

Risk Classification

High: Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, of the contract and its functions. Must be fixed as soon as possible.

Medium: Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Must be fixed as soon as possible.

Low: Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.

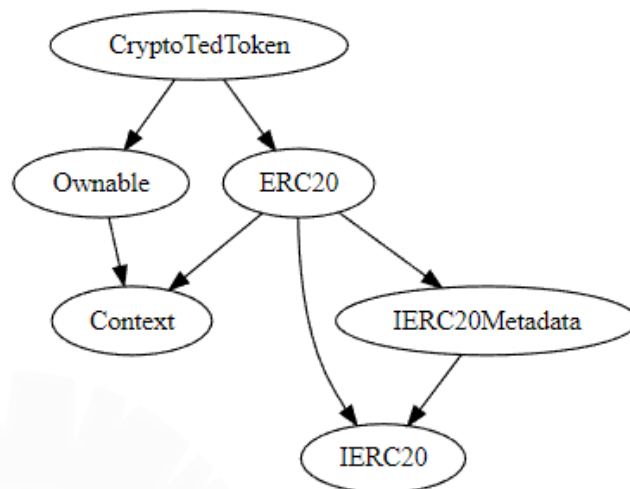
Automated Analysis

Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
Context	Implementation			
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		
Ownable	Implementation	Context		
L		Public !	●	NO !
L	owner	Public !		NO !
L	_checkOwner	Internal 🔒		
L	renounceOwnership	Public !	●	onlyOwner
L	transferOwnership	Public !	●	onlyOwner
L	_transferOwnership	Internal 🔒	●	
IERC20	Interface			
L	totalSupply	External !		NO !
L	balanceOf	External !		NO !
L	transfer	External !	●	NO !
L	allowance	External !		NO !
L	approve	External !	●	NO !
L	transferFrom	External !	●	NO !
IERC20Metadata	Interface	IERC20		
L	name	External !		NO !
L	symbol	External !		NO !
L	decimals	External !		NO !
ERC20	Implementation	Context, IERC20, IERC20Metadata		
L		Public !	●	NO !
L	name	Public !		NO !
L	symbol	Public !		NO !
L	decimals	Public !		NO !
L	totalSupply	Public !		NO !
L	balanceOf	Public !		NO !

Automated Analysis

L	transfer	Public !	●	NO !
L	allowance	Public !		NO !
L	approve	Public !	●	NO !
L	transferFrom	Public !	●	NO !
L	increaseAllowance	Public !	●	NO !
L	decreaseAllowance	Public !	●	NO !
L	_transfer	Internal 🔒	●	
L	_mint	Internal 🔒	●	
L	_burn	Internal 🔒	●	
L	_approve	Internal 🔒	●	
L	_spendAllowance	Internal 🔒	●	
L	_beforeTokenTransfer	Internal 🔒	●	
L	_afterTokenTransfer	Internal 🔒	●	
CryptoTedToken	Implementation	ERC20, Ownable		
L		Public !	●	ERC20
L	setPair	External !	●	onlyOwner
L	setLockLiquidity	External !	●	onlyOwner
L	SetMultiSigAuthority	External !	●	onlyOwner
L	SetVesting	External !	●	onlyOwner
L	transfer	Public !	●	NO !
L	transferFrom	Public !	●	NO !
L	renounceOwnership	Public !	●	onlyOwner
L	antiBotMechanism	Public !		NO !

Inheritance Graph





Components

 Contracts	 Libraries	 Interfaces	 Abstract
2	0	2	2

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.





 Public	 Payable
31	0







External	Internal	Private	Pure	View
13	39	0	1	16


StateVariables

Total	 Public
10	4

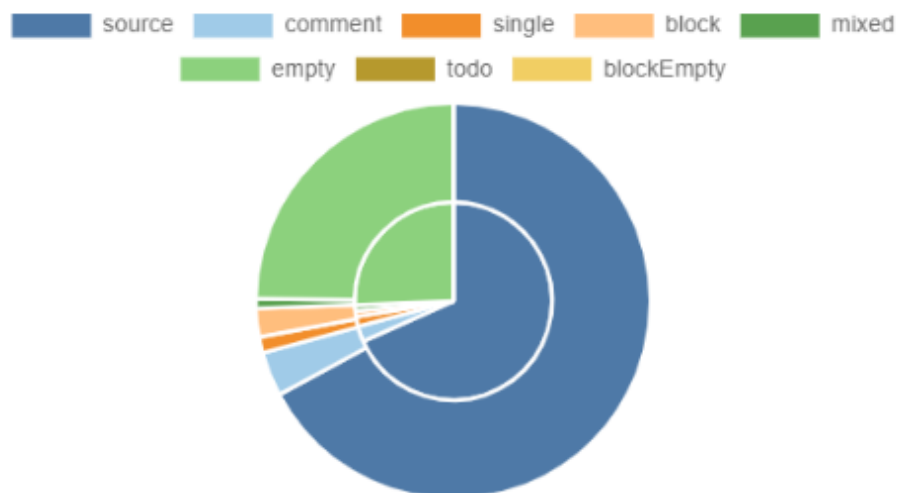
Capabilities

Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
<input type="text" value="^0.8.0"/>		<input type="text"/>	<input type="text"/>	<input type="text"/>

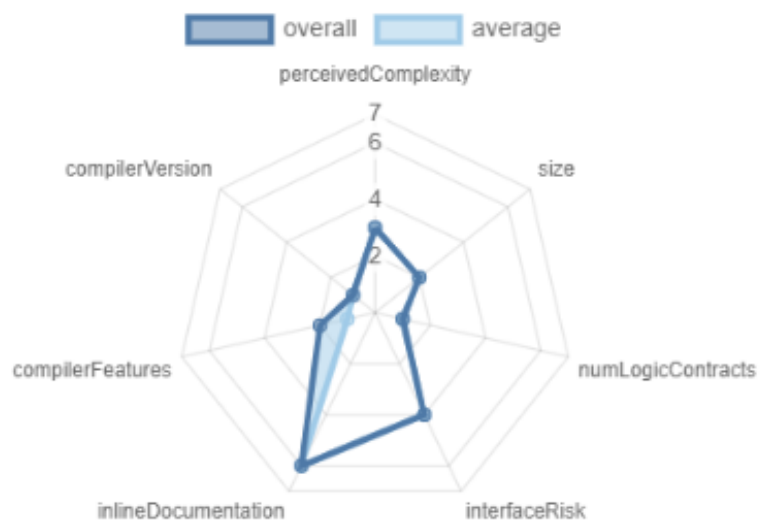
 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECRecover	 New/Create/Create2
<input type="text" value="yes"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

 TryCatch	Σ Unchecked
<input type="text"/>	<input type="text" value="yes"/>

Source Lines



Risk



Manual Review

The owner can pause trading

```
function setLockLiquidity(uint256 _lockLiquidity) external onlyOwner {  
    LockLiquidity = _lockLiquidity;  
}
```

Description

setLockLiquidity function allows the contract owner to set the value of the LockLiquidity variable. The `_lockLiquidity` parameter represents a Unix timestamp. The function can only be called by the contract owner due to the `onlyOwner` modifier. but owner can use this function like pause and unpause function

Recommendation

Implement proper input validation and sanity checks for the `_lockLiquidity` parameter in the `setLockLiquidity` function. Ensure that the new lock liquidity value is valid and does not allow for undesirable or incorrect behavior. Trading must be enabled by the owner just one time can't change the status(pause/unpause).

Manual Review

The owner can add/remove whitelisted address

```
function SetMultiSigAuthority(address _address) external onlyOwner {  
    whitelistedAddresses[_address] = true;  
}  
  
function SetVesting(address _address) external onlyOwner {  
    whitelistedAddresses[_address] = false;  
}
```

Description

SetMultiSigAuthority function allows the contract owner to add an address to the whitelistedAddresses mapping with a value of true. **SetVesting** function allows the contract owner to add an address to the whitelistedAddresses mapping with a value of false.

Recommendation

You should carefully manage the private key of the owner's account. You should use powerful security mechanism that will prevent a single user from accessing the contract owner functions. That risk can be prevented by temporarily locking the contract or renouncing ownership

Manual Review

The owner can set pair address

```
function setPair(address _pair) external onlyOwner  
    pair = _pair;  
}
```

Description

setPair function allows the contract owner to set the value of the pair variable, representing the UniSwap Pair contract address. Only the contract owner can invoke this function due to the onlyOwner modifier.

Recommendation

You should carefully manage the private key of the owner's account. You should use powerful security mechanism that will prevent a single user from accessing the contract owner functions. Implement proper input validation and sanity checks for the _pair parameter in the setPair function. Ensure that only valid UniSwap Pair contract addresses can be set as the pair variable.



AUDIT REPORT

SecureWise



<https://securewise.info/>



<https://t.me/securewisehub>



<https://github.com/securewise>

