Cyberscope

Audit Report

NOVAWCHI

March 2023

Network	ETH
Address	0x337AF08bb6980Ecb68389C5ed8876D08643aBF8a
Audited by	© cyberscope

Table of Contents

Review

Contract Name	Novawchi
Compiler Version	v0.8.18+commit.87f61d96
Optimization	200 runs
Explorer	https://etherscan.io/address/0x337af08bb6980ecb68389c5ed8876d08643abf8a
Address	0x337af08bb6980ecb68389c5ed8876d08643abf8a
Network	ETH
Symbol	VACHI
Decimals	18
Total Supply	9,666,666

Audit Updates

Initial Audit	29 Mar 2023

Source Files

Filename	SHA256
Novawchi.sol	dd00d1c2bdc4996dd05f96ab81a279d8bcf22900a953c76faabf180a488a2a22

Findings Breakdown

Critical	2
Medium	0
Minor / Informative	6

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	2	0	0	0
Medium	0	0	0	0
Minor / Informative	6	0	0	0

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
	ST	Stops Transactions	Passed
	OCTD	Transfers Contract's Tokens	Passed
	OTUT	Transfers User's Tokens	Passed
	ELFM	Exceeds Fees Limit	Passed
	ULTW	Transfers Liquidity to Team Wallet	Passed
	MT	Mints Tokens	Passed
	BT	Burns Tokens	Unresolved
	BC	Blacklists Addresses	Passed

BT - Burns Tokens

Criticality	Critical
Location	Novawchi.sol#L956
Status	Unresolved

Description

The contract owner has the authority to assign addresses to the safeManager mapping. These addresses are able to burn their own tokens. Additionally, the amount that is subtracted from the sender is not equal to the amount that is added to the burnAddress and subtracted from the total supply. The rAmount variable that is subtracted from the sender and is added to the burnAddress is irrelevant to the amount that the sender wants to burn. More details on the total supply diversion issue can be found at the TSD section.

These addresses may take advantage of it by calling the `burn` function. As a result, the more the `safeManager` addresses are, the more the sum of balances will diverse from the total supply.

```
function _burn(address account, uint256 amount) internal virtual {
    require(account != burnAccount, "ERC20: burn from the burn address");

    uint256 accountBalance = balanceOf(account);
    require(accountBalance >= amount, "ERC20: burn amount exceeds
balance");

    uint256 rAmount = _getRate();

    // Transfer from account to the burnAccount
    _rOwned[account] -= rAmount;

    _tOwned[burnAccount] += amount;
    _rOwned[burnAccount] += rAmount;

    _tTotal -= amount;

    _totalBurnt += amount;

    emit Transfer(account, burnAccount, amount);
}
```

Recommendation

The sum of balances should always equal the total supply.

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
Critical	TSD	Total Supply Diversion	Unresolved
Minor / Informative	L02	State Variables could be Declared Constant	Unresolved
Minor / Informative	L04	Conformance to Solidity Naming Conventions	Unresolved
Minor / Informative	L05	Unused State Variable	Unresolved
Minor / Informative	L11	Unnecessary Boolean equality	Unresolved
Minor / Informative	L13	Divide before Multiply Operation	Unresolved
Minor / Informative	L20	Succeeded Transfer Check	Unresolved

TSD - Total Supply Diversion

Criticality	Critical
Location	Novawchi.sol#L961
Status	Unresolved

Description

The total supply of a token is the total number of tokens that have been created, while the balances of individual accounts represent the number of tokens that an account owns. The total supply and the balances of individual accounts are two separate concepts that are managed by different variables in a smart contract. These two entities should be equal to each other.

In the contract, the amount that is subtracted from the total supply does not equal the amount that is subtracted from the balances. As a result, the sum of balances is diverse from the total supply.

```
function _burn(address account, uint256 amount) internal virtual {
    require(account != burnAccount, "ERC20: burn from the burn address");

    uint256 accountBalance = balanceOf(account);
    require(accountBalance >= amount, "ERC20: burn amount exceeds
balance");

    uint256 rAmount = _getRate();

    // Transfer from account to the burnAccount
    _rOwned[account] -= rAmount;

    _tOwned[burnAccount] += amount;
    _rOwned[burnAccount] += rAmount;

    _tTotal -= amount;

    _totalBurnt += amount;

    emit Transfer(account, burnAccount, amount);
}
```

Recommendation

The total supply and the balance variables are separate and independent from each other. The total supply represents the total number of tokens that have been created, while the balance mapping stores the number of tokens that each account owns. The sum of balances should always equal the total supply.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	Novawchi.sol#L51,52,449,457,458,459,467,470,471,472
Status	Unresolved

Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
address payable private _previousOwner
uint256 private _lockTime
address private burnAccount = 0x0000000000000000000000000000000000000001
string private _name = "Novawchi"
string private _symbol = "VACHI"
uint8 private _decimals = 18
uint256 public maxFee = 1000
uint256 public rewardFee = 750
uint256 public teamFee = 100
uint256 public treasuryFee = 150
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Novawchi.sol#L155,157,188,234,599,705,706,720,730,735
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
function DOMAIN_SEPARATOR() external view returns (bytes32);
function PERMIT_TYPEHASH() external pure returns (bytes32);
function MINIMUM_LIQUIDITY() external pure returns (uint256);
function WETH() external pure returns (address);
address payable _safeManager
bool _value
address payable _teamWallet
address payable _treasuryWallet
IdexRouter02 _router
address _pair
uint256 _amount
IERC20 _token
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.
Find more information on the Solidity documentation
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L05 - Unused State Variable

Criticality	Minor / Informative
Location	Novawchi.sol#L51,52
Status	Unresolved

Description

An unused state variable is a state variable that is declared in the contract, but is never used in any of the contract's functions. This can happen if the state variable was originally intended to be used, but was later removed or never used.

Unused state variables can create clutter in the contract and make it more difficult to understand and maintain. They can also increase the size of the contract and the cost of deploying and interacting with it.

```
address payable private _previousOwner
uint256 private _lockTime
```

Recommendation

To avoid creating unused state variables, it's important to carefully consider the state variables that are needed for the contract's functionality and to remove any that are no longer needed. This can help improve the clarity and efficiency of the contract.

L11 - Unnecessary Boolean equality

Criticality	Minor / Informative
Location	Novawchi.sol#L957
Status	Unresolved

Description

Boolean equality is unnecessary when comparing two boolean values. This is because a boolean value is either true or false, and there is no need to compare two values that are already known to be either true or false.

it's important to be aware of the types of variables and expressions that are being used in the contract's code, as this can affect the contract's behavior and performance. The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
require(safeManager[_msgSender()] == true)
```

Recommendation

Using the boolean value itself is clearer and more concise, and it is generally considered good practice to avoid unnecessary boolean equalities in Solidity code.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	Novawchi.sol#L751,752,778,779,787,788
Status	Unresolved

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause a loss of prediction.

```
uint256 tFee = (tAmount * (_currentRewardFee)) / (10000)
uint256 rFee = tFee * (_getRate())
```

Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	Novawchi.sol#L737
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
_token.transfer(msg.sender, _amount)
```

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner

IUniswapFactory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
IdexPair	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-

	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	mint	External	✓	-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-
	initialize	External	✓	-
IdexRouter01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-

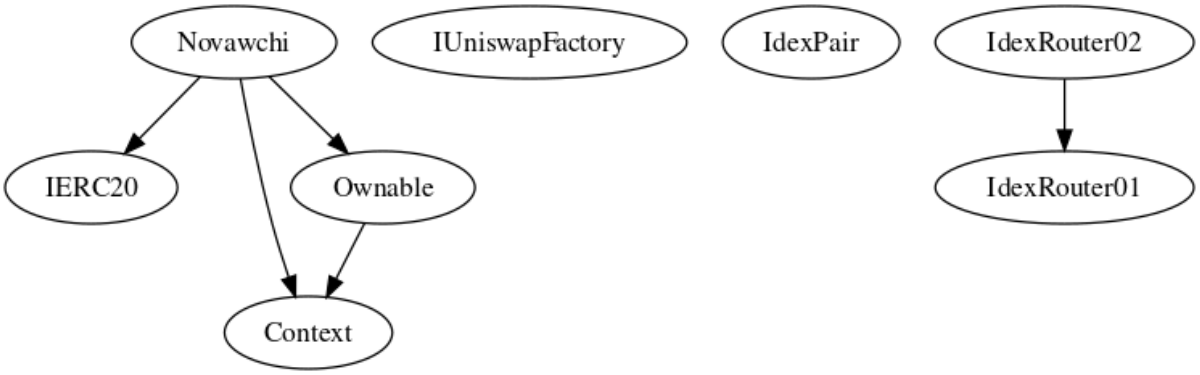
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
IdexRouter02	Interface	IdexRouter01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-

Novawchi	Implementation	Context, IERC20, Ownable		
		Public	✓	-
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	isExcludedFromReward	Public		-
	totalFees	Public		-
	setSafeManager	Public	✓	onlyOwner
	deliver	Public	✓	-
	reflectionFromToken	Public		-
	tokenFromReflection	Public		-
	excludeFromReward	Public	✓	onlyOwner
	excludeMultipleAccountsFromReward	Public	✓	onlyOwner
	includeInReward	External	✓	onlyOwner
	excludeFromFee	Public	✓	onlyOwner

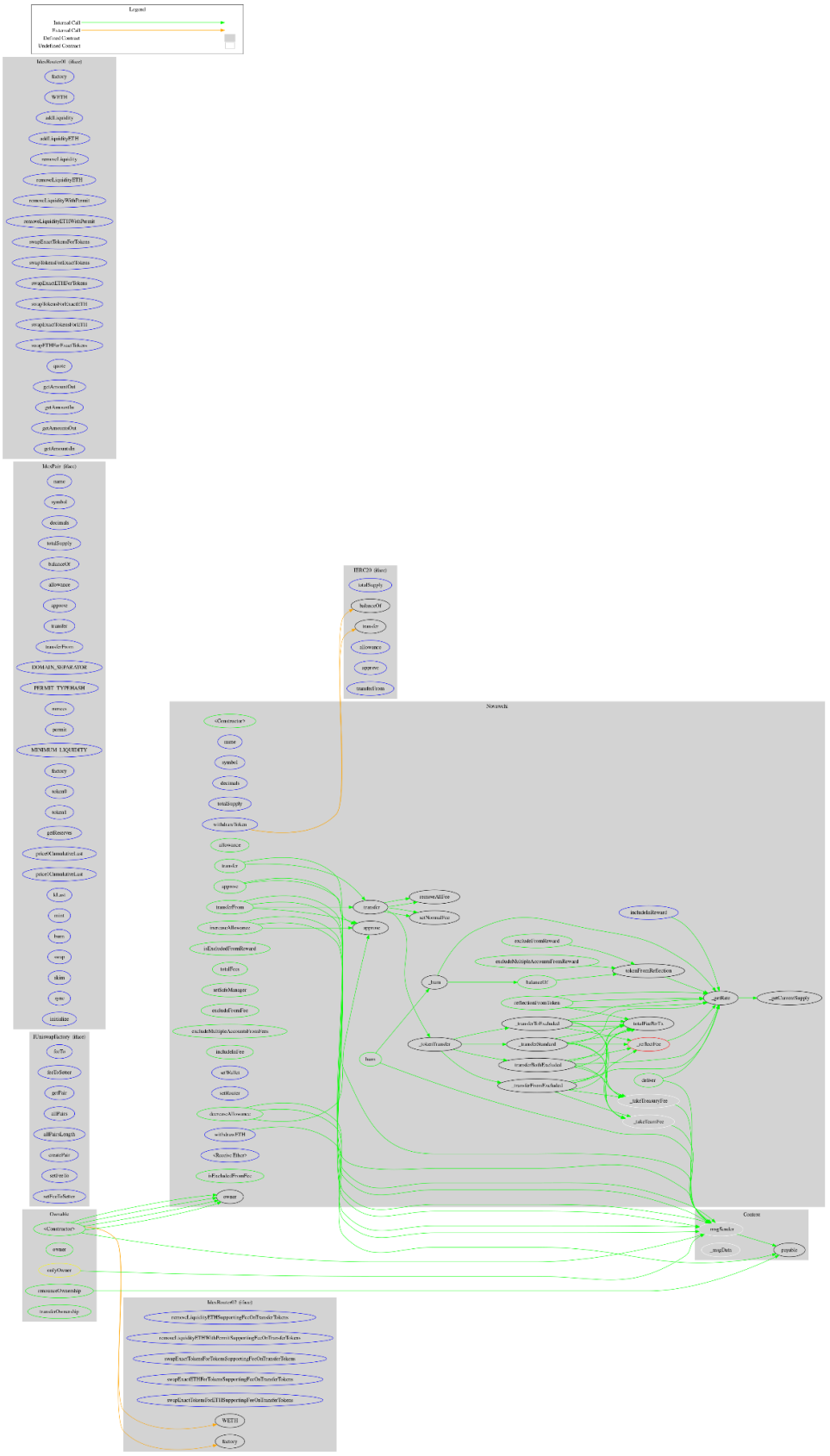
	excludeMultipleAccountsFromFees	Public	✓	onlyOwner
	includeInFee	Public	✓	onlyOwner
	setWallet	External	✓	onlyOwner
	setRouter	External	✓	onlyOwner
	withdrawETH	External	✓	onlyOwner
	withdrawToken	External	✓	onlyOwner
		External	Payable	-
	totalFeePerTx	Internal		
	_reflectFee	Private	✓	
	_getRate	Private		
	_getCurrentSupply	Private		
	_takeTeamFee	Internal	✓	
	_takeTreasuryFee	Internal	✓	
	removeAllFee	Private	✓	
	setNormalFee	Private	✓	
	isExcludedFromFee	Public		-
	_approve	Private	✓	
	_transfer	Private	✓	
	_tokenTransfer	Private	✓	
	_transferStandard	Private	✓	
	_transferToExcluded	Private	✓	
	_transferFromExcluded	Private	✓	
	_transferBothExcluded	Private	✓	

	burn	Public	✓	-
	_burn	Internal	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IUniswapV2Router01 (iface)
factory
WETH
addLiquidity
addLiquidityETH
removeLiquidity
removeLiquidityETH
removeLiquidityWithPermit
removeLiquidityETHWithPermit
swapExactTokensForTokens
swapTokensForExactTokens
swapExactETHForTokens
swapTokensForExactETH
swapExactTokensForETH
swapETHForExactTokens
quote
getAmountOut
getAmountIn
getAmountsOut
getAmountsIn
IUniswapV2Pair (iface)
name
symbol
decimals
totalSupply
balanceOf
allowance
approve
transfer
transferFrom
DOMAIN_SEPARATOR
PERMIT_TYPEHASH
nonces
permit
MINIMUM_LIQUIDITY
factory
token0
token1
getReserves
price0CumulativeLast
price1CumulativeLast
kLast
mint
burn
swap
skim
sync
initialize
IUniswapFactory (iface)
feeTo
feeToSetter
getPair
allPairs
allPairsLength
createPair
setFeeTo
setFeeToSetter
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Nowawchi
<Constructor>
name
symbol
decimals
totalSupply
withdrawToken
allowance
transfer
approve
transferFrom
increaseAllowance
isExcludedFromReward
totalFees
setSwapManager
excludeFromFee
excludeMultipleAccountsFromFees
includeInFee
setWallet
setRouter
decreaseAllowance
withdrawBNB
<Receive Ether>
isExcludedFromFee
owner
removeAllFee
setNormalFee
_transfer
_approve
includeInReward
excludeFromReward
excludeMultipleAccountsFromReward
tokenFromReflection
balanceOf
reflectionFromToken
_getRate
_getCurrentSupply
_transferBothExcluded
_transferStandard
totalFeePerTx
_takeFee
transferBothExcluded
transferFromExcluded
deliver
_takeTeamFee
_getTaxFee
_transferToExcluded
Ownable
<Constructor>
owner
onlyOwner
renounceOwnership
transferOwnership
Context
_msgSender
_msgData
payable
IdexRouter02 (iface)
removeLiquidityETHSupportingFeeOnTransferTokens
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
factory

Summary

NOVAWCHI contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. There are some functions that can be abused by the owner like burn tokens from any address. If the contract owner abuses the burn functionality, then the users could lost their tokens. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats. The fees are fixed at 10% for buys and transfers, and 0% for sales.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/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 and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io