



Cyberscope

# Audit Report

## **Wily Coyote**

July 2023

SHA256      30b98bdc29d3dc8882489a3353f3be58e7968c5ab7999ab2dfde8e3e992e4fd8

Audited by   © cyberscope

# Analysis

● Critical   ● Medium   ● Minor / Informative   ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

# Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	MEE	Missing Events Emission	Unresolved
●	UFM	Unused Fee/Swap Mechanism	Unresolved
●	RCS	Redundant Conditional Statement	Unresolved
●	FSA	Fixed Swap Address	Unresolved
●	PAV	Pair Address Validation	Unresolved
●	RSW	Redundant Storage Writes	Unresolved
●	RES	Redundant Event Statement	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L16	Validate Variable Setters	Unresolved

# Table of Contents

<b>Analysis</b>	<b>1</b>
<b>Diagnostics</b>	<b>2</b>
<b>Table of Contents</b>	<b>3</b>
<b>Review</b>	<b>5</b>
Audit Updates	5
Source Files	5
<b>Findings Breakdown</b>	<b>6</b>
ST - Stops Transactions	7
Description	7
Recommendation	7
MEE - Missing Events Emission	8
Description	8
Recommendation	8
UFM - Unused Fee/Swap Mechanism	9
Description	9
Recommendation	9
RCS - Redundant Conditional Statement	11
Description	11
Recommendation	11
FSA - Fixed Swap Address	12
Description	12
Recommendation	12
PAV - Pair Address Validation	14
Description	14
Recommendation	14
RSW - Redundant Storage Writes	15
Description	15
Recommendation	15
RES - Redundant Event Statement	16
Description	16
Recommendation	16
IDI - Immutable Declaration Improvement	17
Description	17
Recommendation	17
L04 - Conformance to Solidity Naming Conventions	18
Description	18
Recommendation	19
L09 - Dead Code Elimination	20
Description	20

Recommendation	20
L16 - Validate Variable Setters	21
Description	21
Recommendation	21
<b>Functions Analysis</b>	<b>22</b>
<b>Inheritance Graph</b>	<b>26</b>
<b>Flow Graph</b>	<b>27</b>
<b>Summary</b>	<b>28</b>
<b>Disclaimer</b>	<b>29</b>
<b>About Cyberscope</b>	<b>30</b>

## Review

Contract Name	WILY
Testing Deploy	<a href="https://testnet.bscscan.com/address/0x372d511ba3febebb3492a82062da6dfb0af19d62">https://testnet.bscscan.com/address/0x372d511ba3febebb3492a82062da6dfb0af19d62</a>
Symbol	WILY
Decimals	18
Total Supply	1,000,000,000

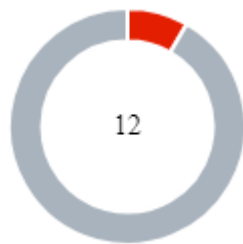
## Audit Updates

Initial Audit	06 Jul 2023
---------------	-------------

## Source Files

Filename	SHA256
contracts/testingDeploy/WILY.sol	30b98bdc29d3dc8882489a3353f3be58e7 968c5ab7999ab2dfde8e3e992e4fd8

## Findings Breakdown



Critical	1
Medium	0
Minor / Informative	11

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	1	0	0	0
Medium	0	0	0	0
Minor / Informative	11	0	0	0

## ST - Stops Transactions

Criticality	Critical
Location	contracts/testingDeploy/WILY.sol#L310
Status	Unresolved

### Description

The transactions are initially disabled for all users excluding the authorized addresses. The owner can enable the transactions for all users. Once the transactions are enable the owner will not be able to disable them again.

```
function _transfer(address from, address to, uint256 amount) internal
returns (bool) {
    bool takeFee = true;
    require(to != address(0), "ERC20: transfer to the zero address");
    require(from != address(0), "ERC20: transfer from the zero address");
    require(amount > 0, "Transfer amount must be greater than zero");

    if (isLimitedAddress(from,to)) {
        require(isTradingEnabled,"Trading is not enabled");
    }
    ...
}
```

### Recommendation

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 multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.



## MEE - Missing Events Emission

<b>Criticality</b>	Minor / Informative
<b>Location</b>	contracts/testingDeploy/WILY.sol#L259
<b>Status</b>	Unresolved

### Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
function setNoFeeWallet(address account, bool enabled) public onlyOwner
{
    _noFee[account] = enabled;
}
```

### Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

## UFM - Unused Fee/Swap Mechanism

Criticality	Minor / Informative
Status	Unresolved

### Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract's fees are set to zero and are immutable. While the contract declares variables and functions related to the fee and swap mechanism, these mechanisms are not utilized within the transaction flow. Consequently, these variables and functions become redundant and serve no purpose within the contract.

```
uint256 constant public swapThreshold = _totalSupply / 20_000;
uint256 constant public buyfee = 0;
uint256 constant public sellfee = 0;
uint256 constant public transferfee = 0;
uint256 constant public fee_denominator = 1_000;
bool private canSwapFees = true;

mapping (address => bool) private isPresaleAddress;

bool private inSwap;

if(is_sell(from, to) && !inSwap && canSwap(from, to)) {
    uint256 contractTokenBalance = balanceOf(address(this));
    if(contractTokenBalance >= swapThreshold) {
        internalSwap(contractTokenBalance); }
}

if (_noFee[from] || _noFee[to]){
    takeFee = false;
}

balance[from] -= amount; uint256 amountAfterFee = (takeFee) ?
takeTaxes(from, is_buy(from, to), is_sell(from, to), amount) : amount;
balance[to] += amountAfterFee; emit Transfer(from, to, amountAfterFee);
```

### Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

## RCS - Redundant Conditional Statement

Criticality	Minor / Informative
Location	contracts/testingDeploy/WILY.sol#L148,149
Status	Unresolved

### Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The variable `_totalSupply` is initialized on the contract's constructor, which means its value is always greater than zero. Hence, the conditional statement is redundant.

```
function totalSupply() external pure override returns (uint256) { if
(_totalSupply == 0) { revert(); } return _totalSupply; }
function decimals() external pure override returns (uint8) { if
(_totalSupply == 0) { revert(); } return _decimals; }
```

### Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

## FSA - Fixed Swap Address

Criticality	Minor / Informative
Location	contracts/testingDeploy/WILY.sol#L204
Status	Unresolved

### Description

The swap address is assigned once and it can not be changed. It is a common practice in decentralized exchanges to create new swap versions. A contract that cannot change the swap address may not be able to catch up to the upgrade. As a result, the contract will not be able to migrate to a new liquidity pool pair or decentralized exchange.

```
constructor () {
    _noFee[msg.sender] = true;

    if (block.chainid == 56) {
        swapRouter =
        IRouter02(0x10ED43C718714eb63d5aA57B78B54704E256024E);
    } else if (block.chainid == 97) {
        swapRouter =
        IRouter02(0xD99D1c33F9fC3444f8101754aBC46c52416550D1);
    } else if (block.chainid == 1 || block.chainid == 4 ||
block.chainid == 3) {
        swapRouter =
        IRouter02(0x7a250d5630B4cF539739dF2C5dAcB4c659F2488D);
    } else if (block.chainid == 43114) {
        swapRouter =
        IRouter02(0x60aE616a2155Ee3d9A68541Ba4544862310933d4);
    } else if (block.chainid == 250) {
        swapRouter =
        IRouter02(0xF491e7B69E4244ad4002BC14e878a34207E38c29);
    } else {
        revert("Chain not valid");
    }
    ...
}
```

### Recommendation

The team is advised to add the ability to change the pair and router address in order to cover potential liquidity pool migrations. It would be better to support multiple pair

addresses so the token will be able to have the same behavior in all the decentralized liquidity pairs.

## PAV - Pair Address Validation

<b>Criticality</b>	Minor / Informative
<b>Location</b>	contracts/testingDeploy/WILY.sol#L292
<b>Status</b>	Unresolved

### Description

The contract is missing address validation in the pair address argument. The absence of validation reveals a potential vulnerability, as it lacks proper checks to ensure the integrity and validity of the pair address provided as an argument. The pair address is a parameter used in certain methods of decentralized exchanges for functions like token swaps and liquidity provisions.

The absence of address validation in the pair address argument can introduce security risks and potential attacks. Without proper validation, if the owner's address is compromised, the contract may lead to unexpected behavior like loss of funds.

```
function changeLpPair(address newPair) external onlyOwner {  
    isLpPair[newPair] = true;  
    emit _changePair(newPair);  
}
```

### Recommendation

To mitigate the risks associated with the absence of address validation in the pair address argument, it is recommended to implement comprehensive address validation mechanisms. A recommended approach could be to verify pair existence in the decentralized application. Prior to interacting with the pair address contract, perform checks to verify the existence and validity of the contract at the provided address. This can be achieved by querying the provider's contract or utilizing external libraries that provide contract verification services.

## RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	contracts/coin/ATM88.sol#L259
Status	Unresolved

### Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract updates state variables even if its current state is the same as the one passed as an argument. As a result, the contract performs redundant storage writes.

```
function setNoFeeWallet(address account, bool enabled) public onlyOwner
{
    _noFee[account] = enabled;
}
```

### Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.



## RES - Redundant Event Statement

Criticality	Minor / Informative
Location	contracts/testingDeploy/WILY.sol#L197
Status	Unresolved

### Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The `_changeThreshold` event statement is not used in the contract's implementation.

```
event _changeThreshold(uint256 newThreshold);
```

### Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it. It is recommend removing the unused event statement from the contract..

## IDI - Immutable Declaration Improvement

<b>Criticality</b>	Minor / Informative
<b>Location</b>	contracts/testingDeploy/WILY.sol#L205,221
<b>Status</b>	Unresolved

### Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
swapRouter  
lpPair
```

### Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

## L04 - Conformance to Solidity Naming Conventions

<b>Criticality</b>	Minor / Informative
<b>Location</b>	contracts/testingDeploy/WILY.sol#L69,178,179,180,193,194,195,196,197,198,271,276,281
<b>Status</b>	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 WETH() external pure returns (address);
string constant private _name = "Wily Coyote"
string constant private _symbol = "WILY"
uint8 constant private _decimals = 18
event _enableTrading();
event _setPresaleAddress(address account, bool enabled);
event _toggleCanSwapFees(bool enabled);
event _changePair(address newLpPair);
event _changeThreshold(uint256 newThreshold);
event _changeWallets(address marketing);

function is_buy(address ins, address out) internal view returns (bool)
{
    bool _is_buy = !isLpPair[out] && isLpPair[ins];
    return _is_buy;
}

...
```

## 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>.

## L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	contracts/testingDeploy/WILY.sol#L281
Status	Unresolved

### Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function is_transfer(address ins, address out) internal view returns
(bool) {
    bool _is_transfer = !isLpPair[out] && !isLpPair[ins];
    return _is_transfer;
}
```

### Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

## L16 - Validate Variable Setters

<b>Criticality</b>	Minor / Informative
<b>Location</b>	contracts/testingDeploy/WILY.sol#L330
<b>Status</b>	Unresolved

### Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
marketingAddress = payable(marketing)
```

### Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

## Functions Analysis

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
<b>Context</b>	Implementation			
		Public	✓	-
	_msgSender	Internal		
	_msgData	Internal		
<b>Ownable</b>	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_setOwner	Private	✓	
<b>IFactoryV2</b>	Interface			
	getPair	External		-
	createPair	External	✓	-
<b>IV2Pair</b>	Interface			
	factory	External		-

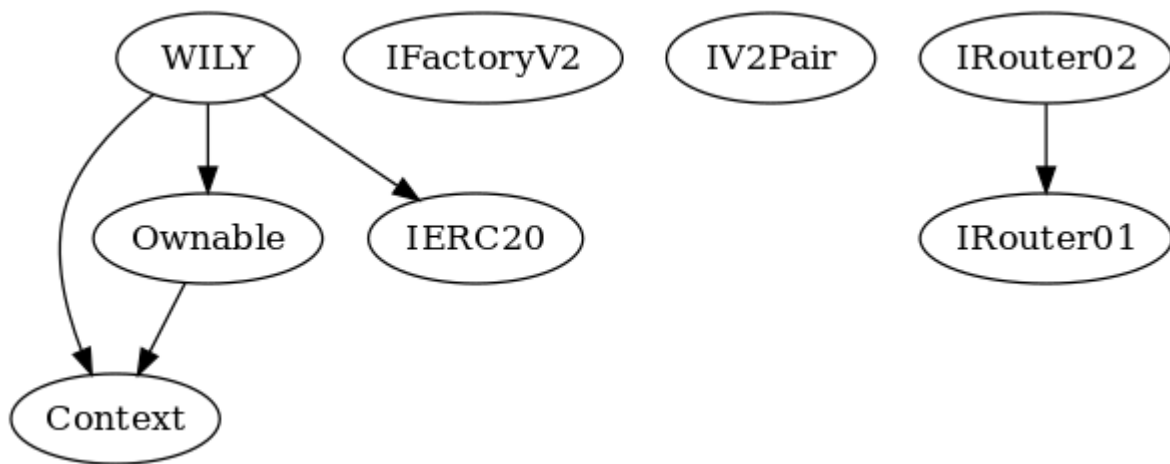
	getReserves	External		-
	sync	External	✓	-
<b>IRouter01</b>	Interface			
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
	addLiquidity	External	✓	-
	swapExactETHForTokens	External	Payable	-
	getAmountsOut	External		-
	getAmountsIn	External		-
<b>IRouter02</b>	Interface	IRouter01		
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokens	External	✓	-
<b>IERC20</b>	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-



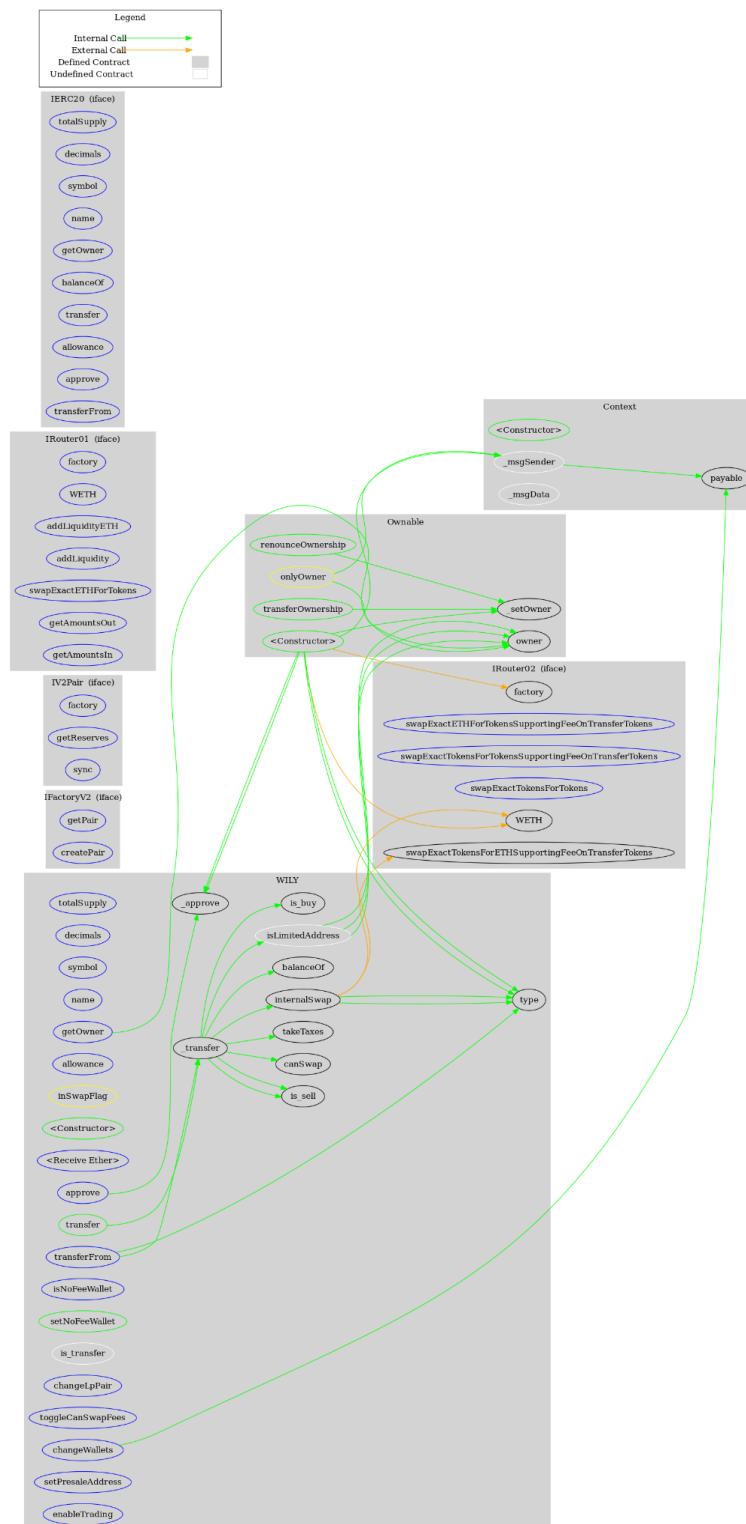
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
<b>WILY</b>	Implementation	Context, Ownable, IERC20		
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	allowance	External		-
	balanceOf	Public		-
		Public	✓	-
		External	Payable	-
	transfer	Public	✓	-
	approve	External	✓	-
	_approve	Internal	✓	
	transferFrom	External	✓	-
	isNoFeeWallet	External		-
	setNoFeeWallet	Public	✓	onlyOwner

	isLimitedAddress	Internal		
	is_buy	Internal		
	is_sell	Internal		
	is_transfer	Internal		
	canSwap	Internal		
	changeLpPair	External	✓	onlyOwner
	toggleCanSwapFees	External	✓	onlyOwner
	_transfer	Internal	✓	
	changeWallets	External	✓	onlyOwner
	takeTaxes	Internal	✓	
	internalSwap	Internal	✓	inSwapFlag
	setPresaleAddress	External	✓	onlyOwner
	enableTrading	External	✓	onlyOwner

## Inheritance Graph



# Flow Graph



## Summary

Wily Coyote 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 stop transactions. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

## 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>