Cyberscope

Audit Report

DEV INU Token

July 2024

Network	BSC
Address	0xdE71903B457f7e23d7f48B9273f79a7e07246882
Audited by	© cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	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
	IDI	Immutable Declaration Improvement	Unresolved
	IECOT	Inefficient ETH Check on Transfers	Unresolved
	IVIU	inSwapAndLiquify Variable Inefficient Usage	Unresolved
	MEM	Misleading Error Messages	Unresolved
	MEE	Missing Events Emission	Unresolved
	PLPI	Potential Liquidity Provision Inadequacy	Unresolved
	PVC	Price Volatility Concern	Unresolved
	RVA	Redundant Variable Assignment	Unresolved
	L02	State Variables could be Declared Constant	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L16	Validate Variable Setters	Unresolved

Table of Contents

Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation**: This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation**: This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical**: Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium**: Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor**: Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative**: Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

Severity	Likelihood / Impact of Exploitation
Critical	Highly Likely / High Impact
Medium	Less Likely / High Impact or Highly Likely/ Lower Impact
Minor / Informative	Unlikely / Low to no Impact

Review

Contract Name	DEVToken
Compiler Version	v0.8.25+commit.b61c2a91
Optimization	200 runs
Explorer	https://bscscan.com/address/0xde71903b457f7e23d7f48b9273f79a7e07246882
Address	0xde71903b457f7e23d7f48b9273f79a7e07246882
Network	BSC
Symbol	DEV
Decimals	18
Total Supply	1,000,000,000
Badge Eligibility	Yes

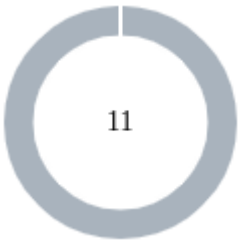
Audit Updates

Initial Audit	15 Jul 2024 https://github.com/cyberscope-io/audits/blob/main/6-dev/v1/audit.pdf
Corrected Phase 2	22 Jul 2024

Source Files

Filename	SHA256
DEVToken.sol	a1b85bd2fe362cb4873028d1cfd8a4b2222b2b5def90675e6c3b3b89b05e1f80

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	11

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

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	DEVToken.sol#L129,130,133
Status	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.

```
buyTax
sellTax
TaxWallet
```

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.

IECOT - Inefficient ETH Check on Transfers

Criticality	Minor / Informative
Location	DEVToken.sol#L319
Status	Unresolved

Description

The contract is currently performing an Ethereum (ETH) balance check on every transaction to determine if it has ETH available to send to the `TaxWallet`. This approach is inefficient because the contract already includes a swap functionality where tokens are exchanged for ETH. By executing this balance check on each transaction, regardless of whether a swap has occurred, the contract incurs unnecessary gas costs, which could negatively impact overall transaction efficiency and cost.

```
    } else if (to == uniswapV2Pair) {
        ...
            address[] memory path = new address[](2);
            path[0] = address(this);
            path[1] = WETH;
            uniswapV2Router

.swapExactTokensForETHSupportingFeeOnTransferTokens(
                    tokensToSwap,
                    0,
                    path,
                    address(this),
                    block.timestamp
                );
            inSwapAndLiquify = 0;
        }
...
uint256 amountReceived = address(this).balance;
uint256 amountETHMarketing = amountReceived;
if (amountETHMarketing > 0)
    transferToAddressETH(TaxWallet, amountETHMarketing);
```

Recommendation

It is recommended to optimize the check for ETH balance by integrating it directly within the swap function block that converts tokens to ETH. This adjustment ensures that the ETH balance is checked and transfered only when new ETH is actually acquired through the swap operation. This would not only streamline the contract's operations by reducing redundant balance checks but also lower the gas consumption, enhancing the contract's performance and reducing operational costs.

IVIU - inSwapAndLiquify Variable Inefficient Usage

Criticality	Minor / Informative
Location	DEVToken.sol#L
Status	Unresolved

Description

The contract is using the `inSwapAndLiquify` variable, which is set only to the values zero and one, to determine if the contract should perform a swap functionality. This implementation is inefficient as the variable essentially acts as a boolean flag but is implemented as an integer. Using an integer variable for boolean logic can lead to unnecessary complexity and potential confusion.

```
if (inSwapAndLiquify == 1) {
    //No tax transfer
    ...
}
if (from == uniswapV2Pair) {
    _tax = buyTax;
    require(balanceOf(to) + (amount) <= maxWalletlimit);
} else if (to == uniswapV2Pair) {
    uint256 tokensToSwap = _balance[address(this)];
    if (tokensToSwap > minSwap && inSwapAndLiquify == 0) {
        ...
        inSwapAndLiquify = 1;
        ...
        inSwapAndLiquify = 0;
    }
```

Recommendation

It is recommended to refactor the contract to use a boolean variable instead of an integer for the `inSwapAndLiquify` flag. This will improve code readability, reduce potential for errors, and optimize the gas usage during contract execution.

MEM - Misleading Error Messages

Criticality	Minor / Informative
Location	DEVToken.sol#L55,73,265
Status	Unresolved

Description

The contract is using misleading error messages. These error messages do not accurately reflect the problem, making it difficult to identify and fix the issue. As a result, the users will not be able to find the root cause of the error.

```
require(isOwner())
require(newOwner != address(0))
require(balanceOf(to) + (amount) <= maxWalletlimit)
```

Recommendation

The team is suggested to provide a descriptive message to the errors. This message can be used to provide additional context about the error that occurred or to explain why the contract execution was halted. This can be useful for debugging and for providing more information to users that interact with the contract.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	DEVToken.sol#L221,225,229,233
Status	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 ExcludeFromFees(address holder, bool exempt) public
onlyOwner {
        _isExcludedFromFees[holder] = exempt;
    }

    function updateSwapandLiquifyLimitonly(bool value) public
onlyOwner {
        swapAndLiquifyByLimitOnly = value;
    }

    function AddLPAddress(address addLPAddress) external
onlyOwner {
        uniswapV2Pair = addLPAddress;
    }

    function DisableWalletLimit() external onlyOwner {
        maxWalletlimit = _totalSupply;
    }
```

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.

PLPI - Potential Liquidity Provision Inadequacy

Criticality	Minor / Informative
Location	DEVToken.sol#L272
Status	Unresolved

Description

The contract operates under the assumption that liquidity is consistently provided to the pair between the contract's token and the native currency. However, there is a possibility that liquidity is provided to a different pair. This inadequacy in liquidity provision in the main pair could expose the contract to risks. Specifically, during eligible transactions, where the contract attempts to swap tokens with the main pair, a failure may occur if liquidity has been added to a pair other than the primary one. Consequently, transactions triggering the swap functionality will result in a revert.

```
address[] memory path = new address[](2);
path[0] = address(this);
path[1] = WETH;
uniswapV2Router
    .swapExactTokensForETHSupportingFeeOnTransferTokens(
        tokensToSwap,
        0,
        path,
        address(this),
        block.timestamp
    );
```

Recommendation

The team is advised to implement a runtime mechanism to check if the pair has adequate liquidity provisions. This feature allows the contract to omit token swaps if the pair does not have adequate liquidity provisions, significantly minimizing the risk of potential failures.

Furthermore, the team could ensure the contract has the capability to switch its active pair in case liquidity is added to another pair.

Additionally, the contract could be designed to tolerate potential reverts from the swap functionality, especially when it is a part of the main transfer flow. This can be achieved by executing the contract's token swaps in a non-reversible manner, thereby ensuring a more resilient and predictable operation.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	DEVToken.sol#L263
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `tokensToSwap` sets a threshold where the contract will trigger the swap functionality. If the variable is set to a big number, then the contract will swap a huge amount of tokens for ETH.

It is important to note that the price of the token representing it, can be highly volatile. This means that the value of a price volatility swap involving Ether could fluctuate significantly at the triggered point, potentially leading to significant price volatility for the parties involved.

```
uint256 tokensToSwap = _balance[address(this)];
if (tokensToSwap > minSwap && inSwapAndLiquify == 0) {
    if (swapAndLiquifyByLimitOnly) {
        tokensToSwap = minSwap;
    } else {
        tokensToSwap = _balance[address(this)];
    }
    inSwapAndLiquify = 1;
    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = WETH;
    uniswapV2Router
        .swapExactTokensForETHSupportingFeeOnTransferTokens(
            tokensToSwap,
            0,
            path,
            address(this),
            block.timestamp
        );
```

Recommendation

The contract could ensure that it will not sell more than a reasonable amount of tokens in a single transaction. A suggested implementation could check that the maximum amount should be less than a fixed percentage of the exchange reserves. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

RVA - Redundant Variable Assignment

Criticality	Minor / Informative
Location	DEVToken.sol#L31
Status	Unresolved

Description

The contract is currently assigning the balance of the contract to `amountReceived`, and then immediately re-assigning the same value to `amountETHMarketing` without modifying it in between these steps. This redundancy in variable assignment is unnecessary and adds to the complexity and gas cost of the contract execution without providing any functional benefit.

```
uint256 amountReceived = address(this).balance;
uint256 amountETHMarketing = amountReceived;
```

Recommendation

It is recommended to eliminate the redundant variable `amountETHMarketing` and use `amountReceived` directly in subsequent operations. This simplification will not only make the code cleaner and easier to understand but also optimize gas usage by reducing the number of storage operations performed during the execution of the contract.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	DEVToken.sol#L104
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.

```
uint256 public minSwap = 500_000 * 10**18
```

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	DEVToken.sol#L96,101,102,103,106,110,111,221,229,233
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 WETH() external pure returns (address);
string private constant _name = "DEV INU TOKEN"
string private constant _symbol = "DEV"
uint256 private constant _totalSupply = 1_000_000_000 * 10**18
uint8 private constant _decimals = 18
address immutable WETH
address payable public TaxWallet

function ExcludeFromFees(address holder, bool exempt) public
onlyOwner {
        _isExcludedFromFees[holder] = exempt;
    }

function AddLPAddress(address addLPAddress) external onlyOwner
{
        uniswapV2Pair = addLPAddress;
    }

...
```

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.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	DEVToken.sol#L230
Status	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.

```
uniswapV2Pair = addLPAddress
```

Recommendation

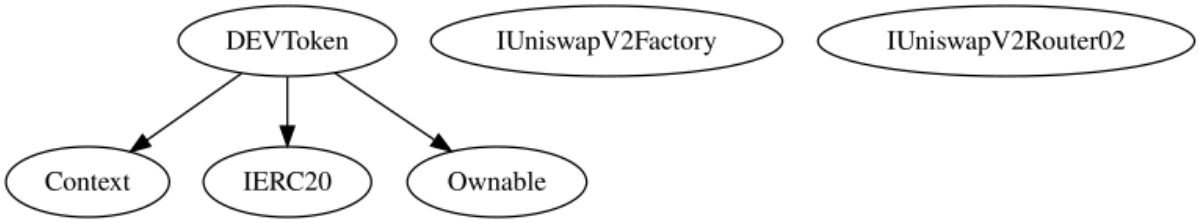
By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

Functions Analysis

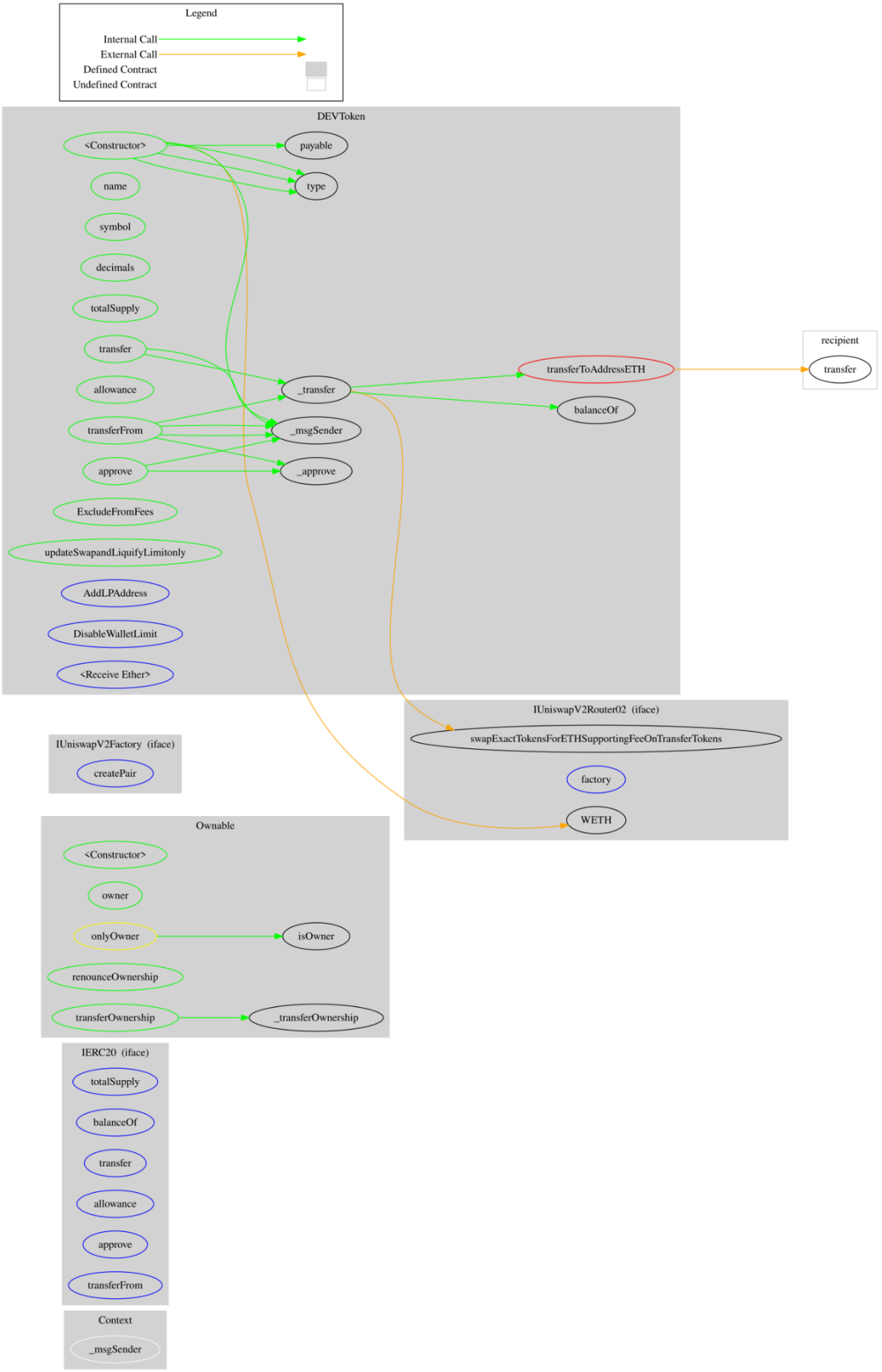
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
DEVToken	Implementation	Context, IERC20, Ownable		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	_approve	Private	✓	
	ExcludeFromFees	Public	✓	onlyOwner
	updateSwapandLiquifyLimitonly	Public	✓	onlyOwner
	AddLPAddress	External	✓	onlyOwner
	DisableWalletLimit	External	✓	onlyOwner
	transferToAddressETH	Private	✓	
	_transfer	Private	✓	

		External	Payable	-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
DEVToken
<Constructor>
payable
type
name
symbol
decimals
totalSupply
transfer
allowance
transferFrom
approve
ExcludeFromFees
updateSwapandLiquifyLimitonly
AddLPAddress
DisableWalletLimit
<Receive Ether>
_transfer
_msgSender
_approve
transferToAddressETH
balanceOf
recipient
transfer
IUniswapV2Router02 (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
factory
WETH
IUniswapV2Factory (iface)
createPair
Ownable
<Constructor>
owner
onlyOwner
isOwner
renounceOwnership
transferOwnership
_transferOwnership
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender

Summary

DEV INU TOKEN contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. DEV INU TOKEN is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler errors or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. There is also a 2% fee on buy and sell transactions.

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

cyberscope.io