Cyberscope

Audit Report

The Worldwide Token

July 2023

SHA256 20de4b252d5de581fc93471b58eec114c09f4ddb81dc29b4ae4715119e6e670e

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	Unresolved
●	MT	Mints Tokens	Unresolved
●	BT	Burns Tokens	Unresolved
●	BC	Blacklists Addresses	Unresolved

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	ZD	Zero Division	Unresolved
●	TFD	Transfer Functions Distinction	Unresolved
●	PSU	Potential Subtraction Underflow	Unresolved
●	MMN	Misleading Method Naming	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L11	Unnecessary Boolean equality	Unresolved
●	L13	Divide before Multiply Operation	Unresolved

Table of Contents

Review

Contract Name	BEP20Token
Testing Deploy	https://testnet.bscscan.com/address/0xe7ecdcc3b925d7ab70d0b1c708498c03eb5d5ecd
Symbol	WORLD
Decimals	4

Audit Updates

Initial Audit	07 Jul 2023 https://github.com/cyberscope-io/audits/tree/main/1-world/v1/audit.pdf
Corrected Phase 2	13 Jul 2023

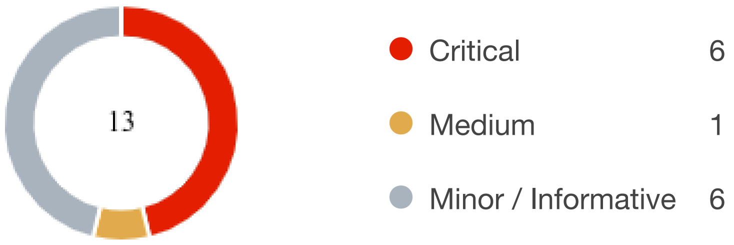
Source Files

Filename	SHA256
contracts/WorldToken_(V3).sol	20de4b252d5de581fc93471b58eec114c09f4ddb81dc29b4ae4715119e6e670e

Overview

The audit scope is to check for security vulnerabilities, validate the business logic and propose potential optimizations. The contract is missing the fundamental principles of a Solidity smart contract regarding gas consumption, code readability, and data structures. According to the previously mentioned issues, the contract cannot be assumed that it is in a production-ready state. Given these issues, it is not advisable to assume that the contract is in a production-ready state. The development team is strongly encouraged to re-evaluate the business logic and Solidity guidelines to ensure that the contract adheres to established best practices and security measures. It is recommended that the team review the contract's gas consumption and optimize it accordingly to minimize costs and improve the contract's efficiency. The code's readability should also be improved by simplifying function definitions and using descriptive variable names, as this will enhance the contract's auditability and maintenance.

Findings Breakdown



Severity	Unresolved	Acknowledged	Resolved	Other
● Critical	6	0	0	0
● Medium	1	0	0	0
● Minor / Informative	6	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	contracts/WorldToken_(V3).sol#L664
Status	Unresolved

Description

The contract implements a fee mechanism which can be disabled. Once disabled, the owner cannot enable it again. The fee amount is splitted into several portions and transferred to their corrensponding recipient. However, the contract executes certain operations to calculate the fee percentage, and each of these operations results in a percentage greater than the percentage denominator. As a result, the majority of the transactions will revert.

```
function transfer(address recipient, uint256 amount) external returns
(bool) {
  if(isInBlockList(msg.sender) == false){
    uint256 transferAmount = amount;

    if(taxFree == 0){
      ...
    }

    _transfer(_msgSender(), recipient, transferAmount);
    return true;
  }
}
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so that users can interact with the contract without interruptions. Some possible solutions could be to:

- disable the fees so that transactions proceed as expected
- completely remove the fee functionality from the contract
- rewrite the fee functionality from the ground up.

ELFM - Exceeds Fees Limit

Criticality	Critical
Location	contracts/WorldToken_(V3).sol#L397,402,407
Status	Unresolved

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling either the `changeTax`, `changeBuyTax`, or `changeTransferTax` function with a high percentage value.

```
function changeTax(uint256 newTax) public onlyOwner returns (bool) {
    tax = newTax;
    return true;
}

function changeBuyTax(uint256 newTax) public onlyOwner returns (bool) {
    buyTax = newTax;
    return true;
}

function changeTransferTax(uint256 newTax) public onlyOwner returns (bool)
{
    transferTax = newTax;
    return true;
}
```

Recommendation

The contract could embody a check for the maximum acceptable value. 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.

MT - Mints Tokens

Criticality	Critical
Location	contracts/WorldToken_(V3).sol#L885,893
Status	Unresolved

Description

The contract owner has the authority to mint tokens. The owner may take advantage of it by calling the `mint` or `mintToAccount` function. As a result, the contract tokens will be highly inflated.

```
function mint(uint256 amount) public onlyOwner returns (bool) {
  _mint(_msgSender(), amount);
  if (!isHolder(_msgSender())) {
      holders.push(_msgSender());
  }
  return true;
}

function mintToAccount(address addr, uint256 amount) public onlyOwner
returns(bool) {
  _mint(address(addr), amount);
  if (!isHolder(addr)) {
      holders.push(addr);
  }
  return true;
}
```

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 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.

BT - Burns Tokens

Criticality	Critical
Location	contracts/WorldToken_(V3).sol#L836
Status	Unresolved

Description

The contract owner has the authority to burn tokens from a specific address. The owner may take advantage of it by calling the `burnFrom` function. As a result, the targeted address will lose the corresponding tokens.

```
function burnFrom(address addr, uint256 amount) public onlyOwner returns
(bool) {
  _burn(addr, amount);
  return true;
}
```

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 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.

BC - Blacklists Addresses

Criticality	Critical
Location	contracts/WorldToken_(V3).sol#L445
Status	Unresolved

Description

The contract owner has the authority to stop addresses from transactions. The owner may take advantage of it by calling the `addToBlockList` function.

```
function addToBlockList(address wallet) public onlyOwner returns (bool) {
  if(isInBlockList(wallet) == false){
     BlockList.push(wallet);
     emit addToBlockListEvent(wallet);
  }
}
```

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 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.

ZD - Zero Division

Criticality	Critical
Location	contracts/WorldToken_(V3).sol#L776
Status	Unresolved

Description

The contract is using variables that may be set to zero as denominators. This can lead to unpredictable and potentially harmful results, such as a transaction revert.

```
uint256 percent = 50*10000/(transferTax*100);
```

Recommendation

It is important to handle division by zero appropriately in the code to avoid unintended behavior and to ensure the reliability and safety of the contract. The contract should ensure that the divisor is always non-zero before performing a division operation. It should prevent the variables to be set to zero, or should not allow the execution of the corresponding statements.

TFD - Transfer Functions Distinction

Criticality	Medium
Location	contracts/WorldToken_(V3).sol#L664,831
Status	Unresolved

Description

The `transfer` and `transferFrom` functions of an ERC20 token are used to transfer tokens from one user to another.The contract implements both functions. However, there is a distinction between the implementation of each function. For instance, the `transferFrom` function only transfers the given amount from the sender to the recipient, while the `transfer` function has additional functionality, like a fee mechanism and allows transaction only if the `msg.sender` is not blacklisted. As a result, the functions implementation is not consistent.

```
function transfer(address recipient, uint256 amount) external returns
(bool) {
  if(isInBlockList(msg.sender) == false){
    uint256 transferAmount = amount;

    if(taxFree == 0){
      ...
    }

    _transfer(_msgSender(), recipient, transferAmount);
    return true;
  }
}

function transferFrom(address sender, address recipient, uint256 amount)
public onlyOwner returns (bool) {
  _transfer(sender, recipient, amount);
  return true;
}
```

Recommendation

The team is advised to ensure that the implementation of the `transfer` and `transferFrom` functions is consistent.

PSU - Potential Subtraction Underflow

Criticality	Minor / Informative
Location	contracts/WorldToken_(V3).sol#L674
Status	Unresolved

Description

The contract subtracts two values, the second value may be greater than the first value if the contract owner misuses the configuration. As a result, the subtraction may underflow and cause the execution to revert.

As part of the transfer flow the contract calculated the convertWorldAmount percentage. However, if the `tax` is less than 10000, the `convertWorldAmount` will be greater than the `transferAmount` .

```
uint256 percent = 7250*10000/tax;
uint256 convertWorldAmount = percent*amount/10000;
transferAmount = transferAmount - convertWorldAmount;
```

Recommendation

The team is advised to properly handle the code to avoid underflow subtractions and ensure the reliability and safety of the contract. The contract should ensure that the first value is always greater than the second value. It should add a sanity check in the setters of the variable or not allow executing the corresponding section if the condition is violated.

MMN - Misleading Method Naming

Criticality	Minor / Informative
Location	contracts/WorldToken_(V3).sol#L424
Status	Unresolved

Description

Methods can have misleading names if their names do not accurately reflect the functionality they contain or the purpose they serve. The contract uses some method names that are too generic or do not clearly convey the underneath functionality. Misleading method names can lead to confusion, making the code more difficult to read and understand. Methods can have misleading names if their names do not accurately reflect the functionality they contain or the purpose they serve. The contract uses some method names that are too generic or do not clearly convey the underneath functionality. Misleading method names can lead to confusion, making the code more difficult to read and understand.

The function `distributeAPY` utilizes a fixed `ApyPerAnnum` value to mint tokens exponentially based on the holders' balances. This functionality has nothing to do with Annual Percentage Yield (APY). As a result, the method's name is misleading.

```
function distributeAPY() public onlyOwner returns (bool)  {
    uint256 apyAmount = 0;
    for (uint256 i = 0; i < holders.length; i++) {
        address holder = holders[i];
        uint256 holderBalance = _balances[holder];
        apyAmount += (holderBalance * ApyPerAnnum) / 1000000;
        if ((_totalSupply + apyAmount) <= maxTotalSupply){
          mintToAccount(holder, apyAmount);
          emit APYDistribution(holder, apyAmount);
        }
    }
    return true;
}
```

Recommendation

It's always a good practice for the contract to contain method names that are specific and descriptive. The team is advised to keep in mind the readability of the code.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/WorldToken_(V3).sol#L360,367,368,379,380,381
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.

```
address[] public BlockList
uint256 public constant maxTotalSupply = 100000000000
uint256 public ApyPerAnnum = 21
event allowListTransferTaxEvent(address addr);
event addToBlockListEvent(address addr);
event addToDexAddressListEvent(address addr);
```

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/WorldToken_(V3).sol#L989
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 _burnFrom(address account, uint256 amount) internal {
      _burn(account, amount);
  }
```

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.

L11 - Unnecessary Boolean equality

Criticality	Minor / Informative
Location	contracts/WorldToken_(V3).sol#L439,446,453,665,771
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.

```
isInTransferAllowList(wallet) == false
isInBlockList(wallet) == false
isInDexAddressList(wallet) == false
isInBlockList(msg.sender) == false
isInTransferAllowList(msg.sender) == false
```

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	contracts/WorldToken_(V3).sol#L640,641,672,673,677,678,682,683,687,688,692,693,704,705,706,710,711,712,716,717,718,722,723,724,737,738,739,743,744,745,749,750,751,755,756,757,761,762,763,776,777,778,782,783,784
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 loss of prediction.

```
percent = 125*10000/(tax*100)
percent = percent*10000/tax
```

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.

Functions Analysis

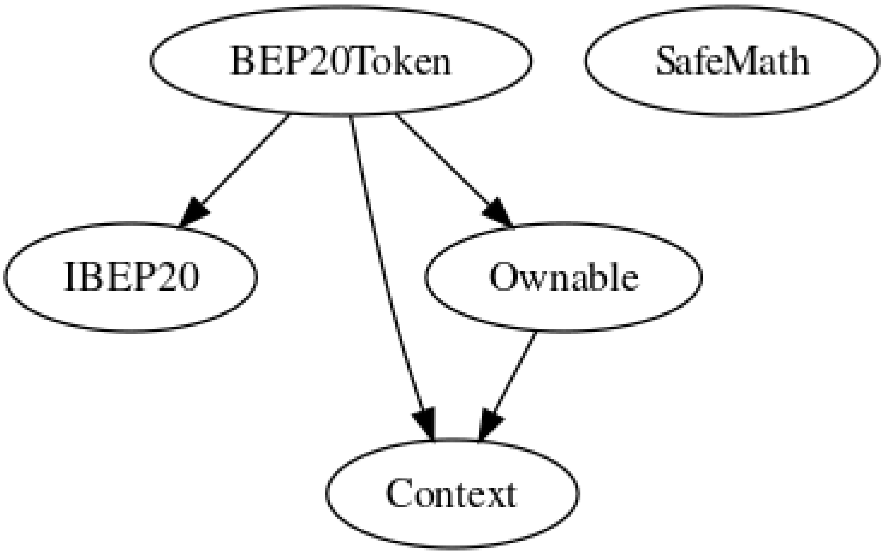
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IBEP20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Context	Implementation			
		Internal	✓	
	_msgSender	Internal		
	_msgData	Internal		
SafeMath	Library			

	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
Ownable	Implementation	Context		
		Internal	✓	
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
BEP20Token	Implementation	Context, IBEP20, Ownable		
		Public	✓	-
	changeTax	Public	✓	onlyOwner
	changeBuyTax	Public	✓	onlyOwner
	changeTransferTax	Public	✓	onlyOwner
	enableTaxFree	Public	✓	onlyOwner
	changeApy	Public	✓	onlyOwner

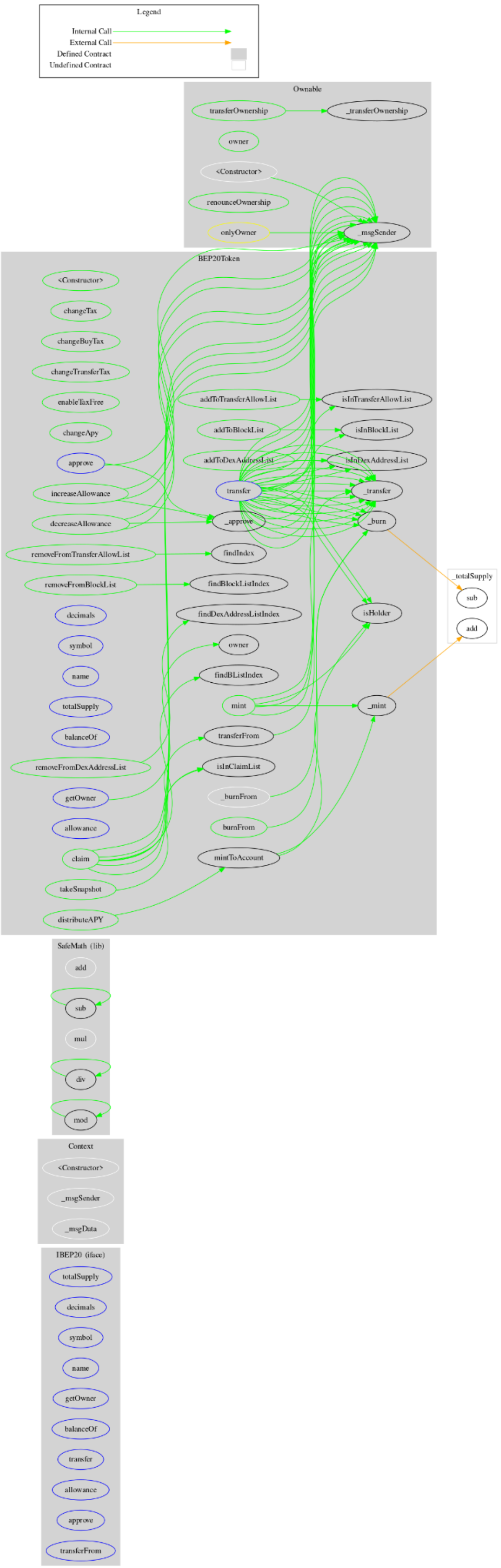
	distributeAPY	Public	✓	onlyOwner
	addToTransferAllowList	Public	✓	onlyOwner
	addToBlockList	Public	✓	onlyOwner
	addToDexAddressList	Public	✓	onlyOwner
	findIndex	Public		-
	findBlockListIndex	Public		-
	findDexAddressListIndex	Public		-
	findBListIndex	Public		-
	removeFromTransferAllowList	Public	✓	onlyOwner
	removeFromBlockList	Public	✓	onlyOwner
	removeFromDexAddressList	Public	✓	onlyOwner
	isInTransferAllowList	Internal		
	isInDexAddressList	Internal		
	isHolder	Internal		
	isInBlockList	Internal		
	isInClaimList	Internal		
	getOwner	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	totalSupply	External		-
	balanceOf	External		-
	takeSnapshot	Public	✓	onlyOwner

	claim	Public	✓	-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	Public	✓	onlyOwner
	burnFrom	Public	✓	onlyOwner
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	mint	Public	✓	onlyOwner
	mintToAccount	Public	✓	onlyOwner
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_burnFrom	Internal	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Ownable
transferOwnership
_transferOwnership
owner
<Constructor>
renounceOwnership
onlyOwner
_msgSender
BEP20Token
<Constructor>
changeTax
changeBuyTax
changeTransferTax
enableTaxFree
changeApy
approve
increaseAllowance
decreaseAllowance
removeFromTransferAllowList
removeFromBlockList
decimals
symbol
name
totalSupply
balanceOf
removeFromDexAddressList
getOwner
allowance
claim
takeSnapshot
distributeAPY
addToTransferAllowList
addToBlockList
addToDexAddressList
transfer
_approve
findIndex
findBlockListIndex
findDexAddressListIndex
owner
findBListIndex
mint
transferFrom
isInClaimList
_burnFrom
burnFrom
mintToAccount
isInTransferAllowList
isInBlockList
isInDexAddressList
_transfer
_burn
isHolder
_mint
_totalSupply
sub
add
SafeMath (lib)
add
sub
mul
div
mod
Context
<Constructor>
_msgSender
_msgData
IBEP20 (iface)
totalSupply
decimals
symbol
name
getOwner
balanceOf
transfer
allowance
approve
transferFrom

Summary

The Worldwide Token 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, manipulating the fees, mint tokens, burning tokens from any address, and massively blacklist addresses. If the contract owner abuses the mint functionality, then the contract will be highly inflated. if the contract owner abuses the burn functionality, then the users could lose 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.

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