Cyberscope

Audit Report

Bulls Run Token

March 2023

Network	BSC
Address	0x615c6AC9e6c3f8a29B41d5Ba4f59B3E3eF268Ee8
Audited by	© cyberscope

Table of Contents

Table of Contents 1
Audit Updates 3
Source Files 3
Analysis 4
ST - Stops Transactions 5
Description 5
Recommendation 5
OTUT - Transfers User's Tokens 6
Description 6
Recommendation 6
ELFM - Exceeds Fees Limit 7
Description 7
Recommendation 8
Diagnostics 9
DDP - Decimal Division Precision 10
Description 10
Recommendation 10
PVC - Price Volatility Concern 11
Description 11
Recommendation 11
IDI - Immutable Declaration Improvement 12
Description 12
Recommendation 12
L02 - State Variables could be Declared Constant 13
Description 13
Recommendation 13
L04 - Conformance to Solidity Naming Conventions 14
Description 14
Recommendation 15
L07 - Missing Events Arithmetic 16
Description 16
Recommendation 16
L16 - Validate Variable Setters 17
Description 17
Recommendation 17
L20 - Succeeded Transfer Check 18
Description 18
Recommendation 18

Functions Analysis 19
Inheritance Graph 23
Flow Graph 24
Summary 25
Disclaimer 26
About Cyberscope 27

Contract Name	BullsRunToken
Compiler Version	v0.7.6+commit.7338295f
Optimization	200 runs
Explorer	https://bscscan.com/address/0x615c6ac9e6c3f8a29b41d5ba4f59b3e3ef268ee8
Address	0x615c6ac9e6c3f8a29b41d5ba4f59b3e3ef268ee8
Network	BSC
Symbol	BULLS RUN
Decimals	9
Total Supply	200,000,000,000

Audit Updates

Initial Audit	15 Mar 2023
Corrected Phase 2	16 Mar 2023
Corrected Phase 3	17 Mar 2023
Corrected Phase 4	17 Mar 2023

Source Files

Filename	SHA256
BullsRunToken.sol	856847544af15cfc9104dc304374f0c5b44a4a402a52e13c60e05379f6f2c356

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

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

ST - Stops Transactions

Criticality	Medium
Location	BullsRunToken.sol#L479
Status	Unresolved

Description

The contract owner has the authority to stop all users from buying excluding the owner. The owner may take advantage of it by setting the `cooldownTimerInterval` to a high value.

```
require(cooldownTimer[recipient] < block.timestamp,"Please wait for 1min between two buys");
cooldownTimer[recipient] = block.timestamp + cooldownTimerInterval;
```

Recommendation

The contract could embody a check for not allowing setting the `cooldownTimerInterval` more than a reasonable amount. 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.

OTUT - Transfers User's Tokens

Criticality	Minor / Informative
Location	BullsRunToken.sol#L687,714
Status	Unresolved

Description

The contract owner has the authority to transfer the balance of a user's address to any address or addresses without allowance. The owner may take advantage of it by calling the `multiTransfer` or `multiTransfer_fixed` functions.

```
function multiTransfer(address from, address[] calldata addresses, uint256[]
calldata tokens) external onlyOwner { ... }

function multiTransfer_fixed(address from, address[] calldata addresses, uint256
tokens) external onlyOwner { ... }
```

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.

ELFM - Exceeds Fees Limit

Criticality	Critical
Location	BullsRunToken.sol#L637
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 the `setFees` and/or `set_sell_multiplier` functions with a high percentage value.

```
function setFees(uint256 _liquidityFee, uint256 _reflectionFee, uint256
_marketingFee, uint256 _feeDenominator) external authorized {
    liquidityFee = _liquidityFee;
    reflectionFee = _reflectionFee;
    marketingFee = _marketingFee;
    devFee = 1;
    totalFee = _liquidityFee.add(_reflectionFee).add(_marketingFee).add(devFee);
    feeDenominator = _feeDenominator;
    require(totalFee < feeDenominator/3, "Fees cannot be more than 25%");
}
...
function set_sell_multiplier(uint256 Multiplier) external onlyOwner{
    sellMultiplier = Multiplier;
}
```

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.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	DDP	Decimal Division Precision	Unresolved
●	PVC	Price Volatility Concern	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

DDP - Decimal Division Precision

Criticality	Minor / Informative
Location	BullsRunToken.sol#L587
Status	Unresolved

Description

Division of decimal (fixed point) numbers can result in rounding errors due to the way that division is implemented in Solidity. Thus, it may produce issues with precise calculations with decimal numbers.

Solidity represents decimal numbers as integers, with the decimal point implied by the number of decimal places specified in the type (e.g. decimal with 18 decimal places). When a division is performed with decimal numbers, the result is also represented as an integer, with the decimal point implied by the number of decimal places in the type. This can lead to rounding errors, as the result may not be able to be accurately represented as an integer with the specified number of decimal places.

Hence, the splitted shares will not have the exact precision and some funds may not be calculated as expected.

```
uint256 amountBNBLiquidity =
amountBNB.mul(dynamicLiquidityFee).div(totalBNBFee).div(2);
uint256 amountBNBReflection = amountBNB.mul(reflectionFee).div(totalBNBFee);
uint256 amountBNBMarketing = amountBNB.mul(marketingFee).div(totalBNBFee);
uint256 amountBNBDev = amountBNB.mul(devFee).div(totalBNBFee);
```

Recommendation

The contract could calculate the subtraction of the divided funds in the last calculation in order to avoid the division rounding issue.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	BullsRunToken.sol#L653
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapThreshold` 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.

```
function setSwapBackSettings(bool _enabled, uint256 _amount) external authorized {
    swapEnabled = _enabled;
    swapThreshold = _amount;
}
```

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 total supply. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	BullsRunToken.sol#L399,400,403
Status	Unresolved

Description

The contract is using variables that initialize them only in the constructor. The other functions are not mutating the variables. These variables are not defined as `immutable`.

```
router
pair
distributor
```

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.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	BullsRunToken.sol#L184,185,198,342,343,344,345,351
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.

```
IBEP20 RWRD = IBEP20(0x2170Ed0880ac9A755fd29B2688956BD959F933F8)
address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
uint256 public dividendsPerShareAccuracyFactor = 10 ** 36
address DEAD = 0x000000000000000000000000000000000000dEaD
address ZERO = 0x0000000000000000000000000000000000000000
address DEV = 0x986aB885cC22Bb4Fc16d0C77FC53039b0B7F6f7f
uint256 _totalSupply = 200000 * 10**6 * 10**_decimals
```

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	BullsRunToken.sol#L120,176,184,185,223,342,343,344,345,347,348,349,351,353,354,356,357,544,549,555,633,643,649,654,659,710
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);
address _token
IBEP20 RWRD = IBEP20(0x2170Ed0880ac9A755fd29B2688956BD959F933F8)
address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
uint256 _minDistribution
uint256 _minPeriod
address DEAD = 0x000000000000000000000000000000000000dEaD
address ZERO = 0x0000000000000000000000000000000000000000
address DEV = 0x986aB885cC22Bb4Fc16d0C77FC53039b0B7F6f7f
string constant _name = "Bulls Run Token"
string constant _symbol = "BULLS RUN"
uint8 constant _decimals = 9
uint256 _totalSupply = 200000 * 10**6 * 10**_decimals
uint256 public _maxTxAmount = _totalSupply

...
```

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.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	BullsRunToken.sol#L224,550,634,651,655
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
minPeriod = _minPeriod
sellMultiplier = Multiplier
liquidityFee = _liquidityFee
swapThreshold = _amount
targetLiquidity = _target
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	BullsRunToken.sol#L106,644,645
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.

```
owner = adr
autoLiquidityReceiver = _autoLiquidityReceiver
marketingFeeReceiver = _marketingFeeReceiver
```

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.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	BullsRunToken.sol#L301
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.

```
RWRD.transfer(shareholder, 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**
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
IBEP20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Auth	Implementation			
		Public	✓	-
	authorize	Public	✓	onlyOwner
	unauthorize	Public	✓	onlyOwner

	isOwner	Public		-
	isAuthorized	Public		-
	transferOwnership	Public	✓	onlyOwner
IDEXFactory	Interface			
	createPair	External	✓	-
IDEXRouter	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IDividendDistributor	Interface			
	setDistributionCriteria	External	✓	-
	setShare	External	✓	-
	deposit	External	Payable	-
	process	External	✓	-
DividendDistributor	Implementation	IDividendDistributor		
		Public	✓	-
	setDistributionCriteria	External	✓	onlyToken
	setShare	External	✓	onlyToken
	deposit	External	Payable	onlyToken
	process	External	✓	onlyToken

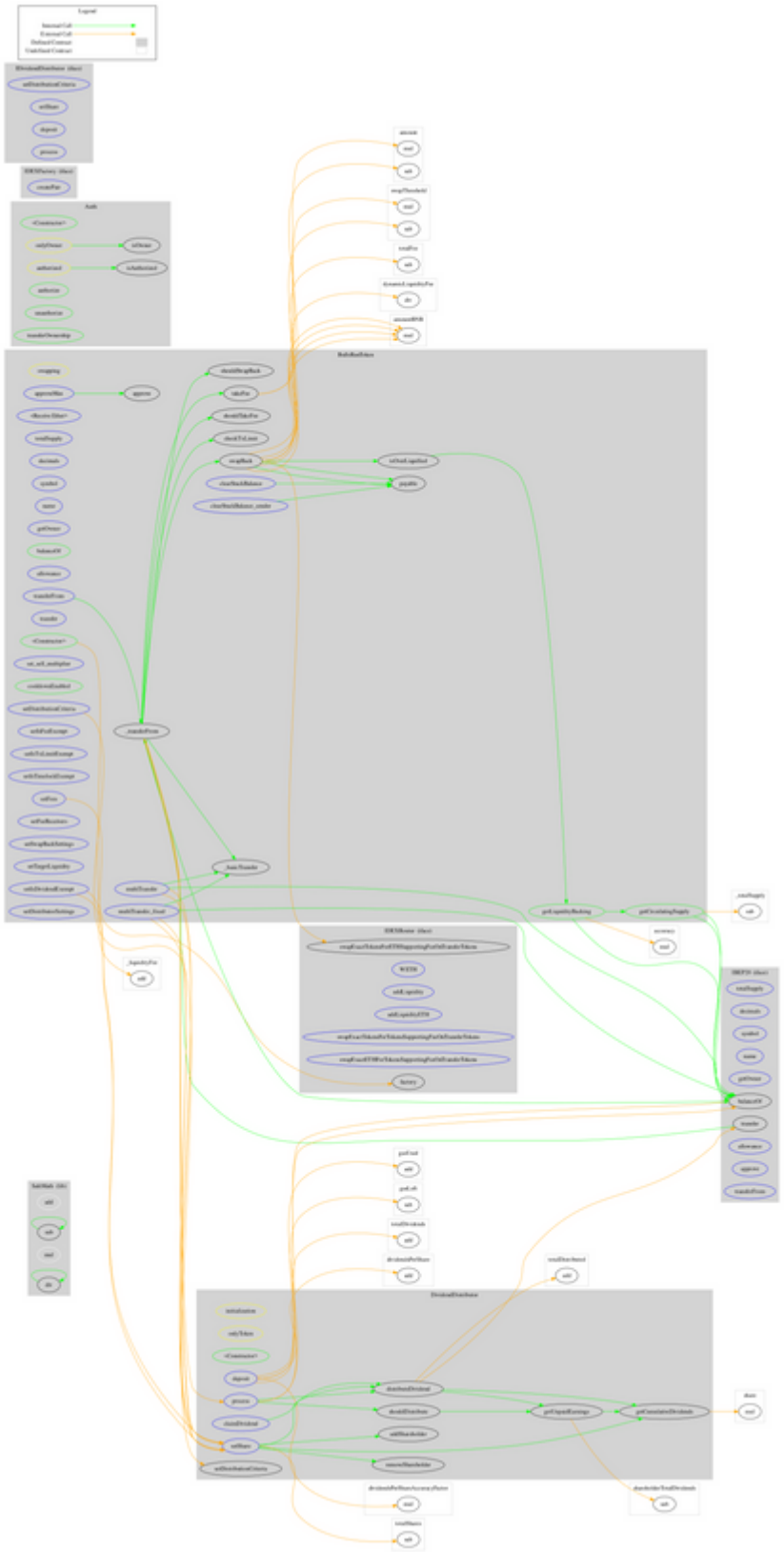
	shouldDistribute	Internal		
	distributeDividend	Internal	✓	
	claimDividend	External	✓	-
	getUnpaidEarnings	Public		-
	getCumulativeDividends	Internal		
	addShareholder	Internal	✓	
	removeShareholder	Internal	✓	
BullsRunToken	Implementation	IBEP20, Auth		
		Public	✓	Auth
		External	Payable	-
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	Public		-
	allowance	External		-
	approve	Public	✓	-
	approveMax	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	_transferFrom	Internal	✓	
	_basicTransfer	Internal	✓	
	checkTxLimit	Internal		
	shouldTakeFee	Internal		
	takeFee	Internal	✓	
	shouldSwapBack	Internal		
	clearStuckBalance	External	✓	authorized

	clearStuckBalance_sender	External	✓	authorized
	set_sell_multiplier	External	✓	onlyOwner
	cooldownEnabled	Public	✓	onlyOwner
	swapBack	Internal	✓	swapping
	setIsDividendExempt	External	✓	authorized
	setIsFeeExempt	External	✓	authorized
	setIsTxLimitExempt	External	✓	authorized
	setIsTimelockExempt	External	✓	authorized
	setFees	External	✓	authorized
	setFeeReceivers	External	✓	authorized
	setSwapBackSettings	External	✓	authorized
	setTargetLiquidity	External	✓	authorized
	setDistributionCriteria	External	✓	authorized
	setDistributorSettings	External	✓	authorized
	getCirculatingSupply	Public		-
	getLiquidityBacking	Public		-
	isOverLiquified	Public		-
	multiTransfer	External	✓	onlyOwner
	multiTransfer_fixed	External	✓	onlyOwner

Inheritance Graph



Flow Graph



Summary

Bulls Run 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 stopping transactions, transferring the user's tokens, and manipulating the fees. 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

Cyberscope

Audit Report

Bulls Run Token

March 2023

Type	BEP20
Network	BSC
Address	0xDEae504798aBC044AAbbCC66709fc6e974088947
Audited by	© cyberscope

Table of Contents

Table of Contents 1
Review 3
Audit Updates 3
Source Files 3
Analysis 4
ST - Stops Transactions 5
Description 5
Recommendation 5
Team Update 6
OTUT - Transfers User's Tokens 7
Description 7
Recommendation 7
Team Update 7
ELFM - Exceeds Fees Limit 8
Description 8
Recommendation 8
Team Update 9
BC - Blacklists Addresses 10
Description 10
Recommendation 10
Team Update 10
Diagnostics 11
DDP - Decimal Division Precision 12
Description 12
Recommendation 12
PVC - Price Volatility Concern 13
Description 13
Recommendation 13
IDI - Immutable Declaration Improvement 14
Description 14
Recommendation 14
L02 - State Variables could be Declared Constant 15
Description 15
Recommendation 15
L04 - Conformance to Solidity Naming Conventions 16
Description 16
Recommendation 17
L07 - Missing Events Arithmetic 18

Description 18
Recommendation 18
L14 - Uninitialized Variables in Local Scope 19
Description 19
Recommendation 19
L16 - Validate Variable Setters 20
Description 20
Recommendation 20
L20 - Succeeded Transfer Check 21
Description 21
Recommendation 21
Functions Analysis 22
Inheritance Graph 26
Flow Graph 27
Summary 28
Disclaimer 29
About Cyberscope 30

Review

Contract Name	BullsRunToken
Compiler Version	v0.7.6+commit.7338295f
Optimization	200 runs
Explorer	https://bscscan.com/address/0xdeae504798abc044aabbcc66709fc6e974088947
Address	0xdeae504798abc044aabbcc66709fc6e974088947
Network	BSC
Symbol	BULLS RUN
Decimals	9
Total Supply	200,000,000,000

Audit Updates

Initial Audit	15 Mar 2023 https://github.com/cyberscope-io/audits/tree/main/bulls-run/v1/audit.pdf
Corrected Phase 2	16 Mar 2023

Source Files

Filename	SHA256
BullsRunToken.sol	f30aba7de9228c6ca4e66e46cd8e0666d34313a6506cd63f00556528cdbaf155

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

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

ST - Stops Transactions

Criticality	Medium
Location	BullsRunToken.sol#L490,495,500
Status	Unresolved

Description

The contract owner has the authority to stop the sales for all users excluding the owner. The owner may take advantage of it by setting the `_maxTxAmount` to zero.

```
if (!authorizations[sender] && recipient != address(this)  && recipient != address(DEAD) && recipient != pair && recipient != marketingFeeReceiver && recipient != devFeeReceiver  && recipient != autoLiquidityReceiver){
    uint256 heldTokens = balanceOf(recipient);
    require((heldTokens + amount) <= _maxWalletToken,"Total Holding is currently limited, you can not buy that much.");}
...
require(cooldownTimer[recipient] < block.timestamp,"Please wait for 1min between two buys");
...
checkTxLimit(sender, amount);
```

Recommendation

The contract could embody a check for not allowing setting the `_maxTxAmount` less than a reasonable amount. A suggested implementation could check that the maximum amount should be more than a fixed percentage of 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.

Team Update

The owner followed our suggestion and renounced ownership at the following transaction: https://bscscan.com/tx/0x8b227e9d3677b6de20cf80367d10d151bc44b1ae3fdd30b700a138f2d9ed6b4b This audit finding remains unresolved, since the `setTxLimit` function is accessible from authorized users, including the owner.

OTUT - Transfers User's Tokens

Criticality	Minor / Informative
Location	BullsRunToken.sol#L716,743
Status	Renounced

Description

The contract owner has the authority to transfer the balance of a user's address to any address or addresses without allowance. The owner may take advantage of it by calling the `multiTransfer` or `multiTransfer_fixed` functions.

```
function multiTransfer(address from, address[] calldata addresses, uint256[]
calldata tokens) external onlyOwner { ... }

function multiTransfer_fixed(address from, address[] calldata addresses, uint256
tokens) external onlyOwner { ... }
```

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.

Team Update

The owner followed our suggestion and renounced ownership at the following transaction:
https://bscscan.com/tx/0x8b227e9d3677b6de20cf80367d10d151bc44b1ae3fdd30b700a138f2d9ed6b4b

ELFM - Exceeds Fees Limit

Criticality	Medium
Location	BullsRunToken.sol#L666
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 the `setFees` function with a high percentage value.

```
function setFees(uint256 _liquidityFee, uint256 _reflectionFee, uint256
_marketingFee, uint256 _feeDenominator) external authorized {
    liquidityFee = _liquidityFee;
    reflectionFee = _reflectionFee;
    marketingFee = _marketingFee;
    devFee = 1;
    totalFee = _liquidityFee.add(_reflectionFee).add(_marketingFee).add(devFee);
    feeDenominator = _feeDenominator;
    require(totalFee < feeDenominator/3, "Fees cannot be more than 33%");
}
```

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.

Team Update

The owner followed our suggestion and renounced ownership at the following transaction: https://bscscan.com/tx/0x8b227e9d3677b6de20cf80367d10d151bc44b1ae3fdd30b700a138f2d9ed6b4b This audit finding remains unresolved, since the `setFees` function is accessible from authorized users, including the owner.

BC - Blacklists Addresses

Criticality	Critical
Location	BullsRunToken.sol#L647
Status	Renounced

Description

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

```
function manage_blacklist(address[] calldata addresses, bool status) public
onlyOwner {
    for (uint256 i; i < addresses.length; ++i) {
        isBlacklisted[addresses[i]] = status;
    }
}
```

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.

Team Update

The owner followed our suggestion and renounced ownership at the following transaction:
https://bscscan.com/tx/0x8b227e9d3677b6de20cf80367d10d151bc44b1ae3fdd30b700a138f2d9ed6b4b

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	DDP	Decimal Division Precision	Unresolved
●	PVC	Price Volatility Concern	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

DDP - Decimal Division Precision

Criticality	Minor / Informative
Location	BullsRunToken.sol#L607
Status	Unresolved

Description

Division of decimal (fixed point) numbers can result in rounding errors due to the way that division is implemented in Solidity. Thus, it may produce issues with precise calculations with decimal numbers.

Solidity represents decimal numbers as integers, with the decimal point implied by the number of decimal places specified in the type (e.g. decimal with 18 decimal places). When a division is performed with decimal numbers, the result is also represented as an integer, with the decimal point implied by the number of decimal places in the type. This can lead to rounding errors, as the result may not be able to be accurately represented as an integer with the specified number of decimal places.

Hence, the splitted shares will not have the exact precision and some funds may not be calculated as expected.

```
uint256 amountBNBLiquidity =
amountBNB.mul(dynamicLiquidityFee).div(totalBNBFee).div(2);
uint256 amountBNBReflection = amountBNB.mul(reflectionFee).div(totalBNBFee);
uint256 amountBNBMarketing = amountBNB.mul(marketingFee).div(totalBNBFee);
uint256 amountBNBDev = amountBNB.mul(devFee).div(totalBNBFee);
```

Recommendation

The contract could calculate the subtraction of the divided funds in the last calculation in order to avoid the division rounding issue.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	BullsRunToken.sol#L682
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapThreshold` 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.

```
function setSwapBackSettings(bool _enabled, uint256 _amount) external authorized {
    swapEnabled = _enabled;
    swapThreshold = _amount;
}
```

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 total supply. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	BullsRunToken.sol#L401,402,405
Status	Unresolved

Description

The contract is using variables that initialize them only in the constructor. The other functions are not mutating the variables. These variables are not defined as `immutable`.

```
route
pai
distributo
```

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.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	BullsRunToken.sol#L184,197,341,342,343,344,350
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 WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
uint256 public dividendsPerShareAccuracyFactor = 10 ** 36
address DEAD = 0x000000000000000000000000000000000000dEaD
address ZERO = 0x0000000000000000000000000000000000000000
address DEV = 0x986aB885cC22Bb4Fc16d0C77FC53039b0B7F6f7f
uint256 _totalSupply = 200000 * 10**6 * 10**_decimals
```

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	BullsRunToken.sol#L119,175,183,184,222,341,342,343,344,346,347,348,350,352,353,355,356,463,466,564,569,574,579,643,647,666,676,682,687,692,743
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);
address _token
IBEP20 RWRD = IBEP20(0x2170Ed0880ac9A755fd29B2688956BD959F933F8)
address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
uint256 _minDistribution
uint256 _minPeriod
address DEAD = 0x000000000000000000000000000000000000dEaD
address ZERO = 0x0000000000000000000000000000000000000000
address DEV = 0x986aB885cC22Bb4Fc16d0C77FC53039b0B7F6f7f
string constant _name = "Bulls Run Token"
string constant _symbol = "BULLS RUN"
uint8 constant _decimals = 9
uint256 _totalSupply = 200000 * 10**6 * 10**_decimals
uint256 public _maxTxAmount = _totalSupply

...
```

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.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	BullsRunToken.sol#L223,467,471,570,667,684,688
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
minPeriod = _minPeriod
_maxTxAmount = (_totalSupply * maxTXPercentage_base1000 ) / 1000
_maxTxAmount = amount
sellMultiplier = Multiplier
liquidityFee = _liquidityFee
swapThreshold = _amount
targetLiquidity = _target
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	BullsRunToken.sol#L648
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
uint256 i
```

Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	BullsRunToken.sol#L105,677,678
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.

```
owner = adr
autoLiquidityReceiver = _autoLiquidityReceiver
marketingFeeReceiver = _marketingFeeReceiver
```

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.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	BullsRunToken.sol#L300
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.

```
RWRD.transfer(shareholder, 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**
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
IBEP20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Auth	Implementation			
		Public	✓	-
	authorize	Public	✓	onlyOwner
	unauthorize	Public	✓	onlyOwner

	isOwner	Public		-
	isAuthorized	Public		-
	transferOwnership	Public	✓	onlyOwner
IDEXFactory	Interface			
	createPair	External	✓	-
IDEXRouter	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IDividendDistributor	Interface			
	setDistributionCriteria	External	✓	-
	setShare	External	✓	-
	deposit	External	Payable	-
	process	External	✓	-
DividendDistributor	Implementation	IDividendDistributor		
		Public	✓	-
	setDistributionCriteria	External	✓	onlyToken
	setShare	External	✓	onlyToken
	deposit	External	Payable	onlyToken
	process	External	✓	onlyToken

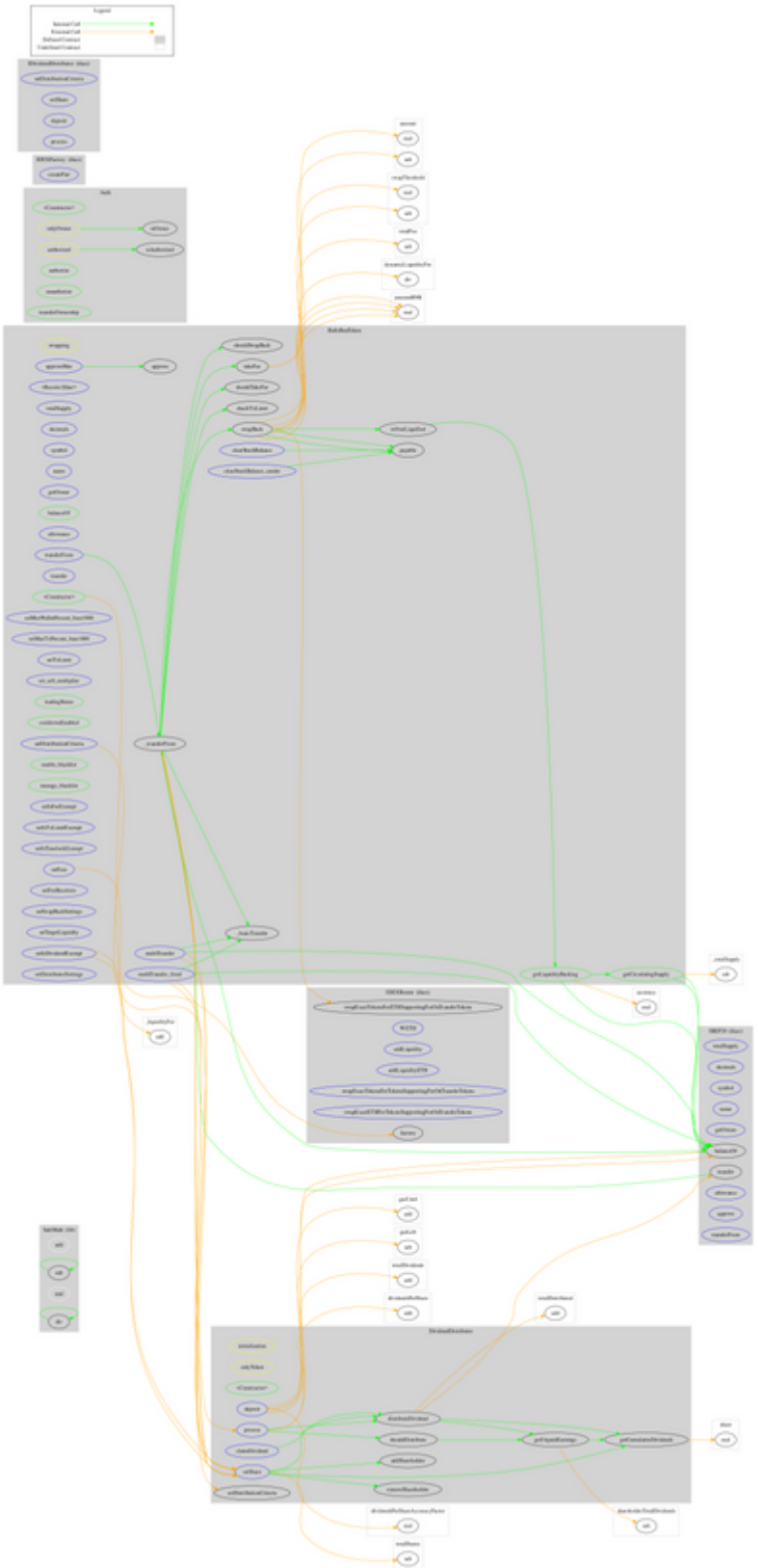
	shouldDistribute	Internal		
	distributeDividend	Internal	✓	
	claimDividend	External	✓	-
	getUnpaidEarnings	Public		-
	getCumulativeDividends	Internal		
	addShareholder	Internal	✓	
	removeShareholder	Internal	✓	
BullsRunToken	Implementation	IBEP20, Auth		
		Public	✓	Auth
		External	Payable	-
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	Public		-
	allowance	External		-
	approve	Public	✓	-
	approveMax	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	setMaxWalletPercent_base1000	External	✓	onlyOwner
	setMaxTxPercent_base1000	External	✓	onlyOwner
	setTxLimit	External	✓	authorized
	_transferFrom	Internal	✓	
	_basicTransfer	Internal	✓	
	checkTxLimit	Internal		
	shouldTakeFee	Internal		

	takeFee	Internal	✓	
	shouldSwapBack	Internal		
	clearStuckBalance	External	✓	authorized
	clearStuckBalance_sender	External	✓	authorized
	set_sell_multiplier	External	✓	onlyOwner
	tradingStatus	Public	✓	onlyOwner
	cooldownEnabled	Public	✓	onlyOwner
	swapBack	Internal	✓	swapping
	setIsDividendExempt	External	✓	authorized
	enable_blacklist	Public	✓	onlyOwner
	manage_blacklist	Public	✓	onlyOwner
	setIsFeeExempt	External	✓	authorized
	setIsTxLimitExempt	External	✓	authorized
	setIsTimelockExempt	External	✓	authorized
	setFees	External	✓	authorized
	setFeeReceivers	External	✓	authorized
	setSwapBackSettings	External	✓	authorized
	setTargetLiquidity	External	✓	authorized
	setDistributionCriteria	External	✓	authorized
	setDistributorSettings	External	✓	authorized
	getCirculatingSupply	Public		-
	getLiquidityBacking	Public		-
	isOverLiquified	Public		-
	multiTransfer	External	✓	onlyOwner
	multiTransfer_fixed	External	✓	onlyOwner

Inheritance Graph



Flow Graph



Summary

Bulls Run Token contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements.

The owner has renounced ownership at the following transaction: https://bscscan.com/tx/0x8b227e9d3677b6de20cf80367d10d151bc44b1ae3fdd30b700a138f2d9ed6b4b.

There are some functions that can still be abused by authorized users, including the owner, like stopping transactions and manipulating the fees. The contract can prevent users from selling if authorized users abuse the admin functions.

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

Cyberscope

Audit Report

Bulls Run Token

March 2023

Github https://github.com/AwakenTraveler/bullsrun

Commit 9c5285698dba11a054661c289454899b94b40300

Audited by © cyberscope

Table of Contents

Table of Contents 1
Review 3
Audit Updates 3
Source Files 3
Analysis 4
ST - Stops Transactions 5
Description 5
Recommendation 5
OTUT - Transfers User's Tokens 6
Description 6
Recommendation 6
ELFM - Exceeds Fees Limit 7
Description 7
Recommendation 8
Diagnostics 9
DDP - Decimal Division Precision 10
Description 10
Recommendation 10
PVC - Price Volatility Concern 11
Description 11
Recommendation 11
IDI - Immutable Declaration Improvement 12
Description 12
Recommendation 12
L02 - State Variables could be Declared Constant 13
Description 13
Recommendation 13
L04 - Conformance to Solidity Naming Conventions 14
Description 14
Recommendation 15
L07 - Missing Events Arithmetic 16
Description 16
Recommendation 16
L16 - Validate Variable Setters 17
Description 17
Recommendation 17
L20 - Succeeded Transfer Check 18
Description 18

Recommendation 18
Functions Analysis 19
Inheritance Graph 23
Flow Graph 24
Summary 25
Disclaimer 26
About Cyberscope 27

Review

Repository	https://github.com/AwakenTraveler/bullsrun
Commit	9c5285698dba11a054661c289454899b94b40300

Audit Updates

Initial Audit	16 Mar 2023 https://github.com/cyberscope-io/audits/tree/main/bulls-run/v1/audit.pdf
Corrected Phase 2	17 Mar 2023

Source Files

Filename	SHA256
BullsRunToken.sol	f30aba7de9228c6ca4e66e46cd8e0666d34313a6506cd63f00556528cdbaf155

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

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

ST - Stops Transactions

Criticality	Medium
Location	BullsRunToken.sol#L475
Status	Unresolved

Description

The contract owner has the authority to stop all users from buying excluding the owner. The owner may take advantage of it by setting the `cooldownTimerInterval` to a high value.

```
require(cooldownTimer[recipient] < block.timestamp,"Please wait for 1min between two buys");
cooldownTimer[recipient] = block.timestamp + cooldownTimerInterval;
```

Recommendation

The contract could embody a check for not allowing setting the `cooldownTimerInterval` more than a reasonable amount. 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.

OTUT - Transfers User's Tokens

Criticality	Minor / Informative
Location	BullsRunToken.sol#L683,710
Status	Unresolved

Description

The contract owner has the authority to transfer the balance of a user's address to any address or addresses without allowance. The owner may take advantage of it by calling the `multiTransfer` or `multiTransfer_fixed` functions.

```
function multiTransfer(address from, address[] calldata addresses, uint256[]
calldata tokens) external onlyOwner { ... }

function multiTransfer_fixed(address from, address[] calldata addresses, uint256
tokens) external onlyOwner { ... }
```

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.

ELFM - Exceeds Fees Limit

Criticality	Critical
Location	BullsRunToken.sol#L633
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 the `setFees` and/or `set_sell_multiplier` functions with a high percentage value.

```
function setFees(uint256 _liquidityFee, uint256 _reflectionFee, uint256
_marketingFee, uint256 _feeDenominator) external authorized {
    liquidityFee = _liquidityFee;
    reflectionFee = _reflectionFee;
    marketingFee = _marketingFee;
    devFee = 1;
    totalFee = _liquidityFee.add(_reflectionFee).add(_marketingFee).add(devFee);
    feeDenominator = _feeDenominator;
    require(totalFee < feeDenominator/3, "Fees cannot be more than 25%");
}
...
function set_sell_multiplier(uint256 Multiplier) external onlyOwner{
    sellMultiplier = Multiplier;
}
```

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.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	DDP	Decimal Division Precision	Unresolved
●	PVC	Price Volatility Concern	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

DDP - Decimal Division Precision

Criticality	Minor / Informative
Location	BullsRunToken.sol#L583
Status	Unresolved

Description

Division of decimal (fixed point) numbers can result in rounding errors due to the way that division is implemented in Solidity. Thus, it may produce issues with precise calculations with decimal numbers.

Solidity represents decimal numbers as integers, with the decimal point implied by the number of decimal places specified in the type (e.g. decimal with 18 decimal places). When a division is performed with decimal numbers, the result is also represented as an integer, with the decimal point implied by the number of decimal places in the type. This can lead to rounding errors, as the result may not be able to be accurately represented as an integer with the specified number of decimal places.

Hence, the splitted shares will not have the exact precision and some funds may not be calculated as expected.

```
uint256 amountBNBLiquidity =
amountBNB.mul(dynamicLiquidityFee).div(totalBNBFee).div(2);
uint256 amountBNBReflection = amountBNB.mul(reflectionFee).div(totalBNBFee);
uint256 amountBNBMarketing = amountBNB.mul(marketingFee).div(totalBNBFee);
uint256 amountBNBDev = amountBNB.mul(devFee).div(totalBNBFee);
```

Recommendation

The contract could calculate the subtraction of the divided funds in the last calculation in order to avoid the division rounding issue.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	BullsRunToken.sol#L649
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapThreshold` 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.

```
function setSwapBackSettings(bool _enabled, uint256 _amount) external authorized {
    swapEnabled = _enabled;
    swapThreshold = _amount;
}
```

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 total supply. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	BullsRunToken.sol#L399,400,403
Status	Unresolved

Description

The contract is using variables that initialize them only in the constructor. The other functions are not mutating the variables. These variables are not defined as `immutable`.

```
router
pair
distributor
```

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.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	BullsRunToken.sol#L185,198,342,343,344,345,351
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 WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
uint256 public dividendsPerShareAccuracyFactor = 10 ** 36
address DEAD = 0x000000000000000000000000000000000000dEaD
address ZERO = 0x0000000000000000000000000000000000000000
address DEV = 0x986aB885cC22Bb4Fc16d0C77FC53039b0B7F6f7f
uint256 _totalSupply = 200000 * 10**6 * 10**_decimals
```

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	BullsRunToken.sol#L120,176,184,185,223,342,343,344,345,347,348,349,351,353,354,356,357,544,549,555,633,643,649,654,659,710
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);
address _token
IBEP20 RWRD = IBEP20(0x2170Ed0880ac9A755fd29B2688956BD959F933F8)
address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
uint256 _minDistribution
uint256 _minPeriod
address DEAD = 0x000000000000000000000000000000000000dEaD
address ZERO = 0x0000000000000000000000000000000000000000
address DEV = 0x986aB885cC22Bb4Fc16d0C77FC53039b0B7F6f7f
string constant _name = "Bulls Run Token"
string constant _symbol = "BULLS RUN"
uint8 constant _decimals = 9
uint256 _totalSupply = 200000 * 10**6 * 10**_decimals
uint256 public _maxTxAmount = _totalSupply

...
```

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.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	BullsRunToken.sol#L224,550,634,651,655
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
minPeriod = _minPeriod
sellMultiplier = Multiplier
liquidityFee = _liquidityFee
swapThreshold = _amount
targetLiquidity = _target
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	BullsRunToken.sol#L106,644,645
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.

```
owner = adr
autoLiquidityReceiver = _autoLiquidityReceiver
marketingFeeReceiver = _marketingFeeReceiver
```

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.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	BullsRunToken.sol#L301
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.

```
RWRD.transfer(shareholder, 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**
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
IBEP20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Auth	Implementation			
		Public	✓	-
	authorize	Public	✓	onlyOwner
	unauthorize	Public	✓	onlyOwner

	isOwner	Public		-
	isAuthorized	Public		-
	transferOwnership	Public	✓	onlyOwner
IDEXFactory	Interface			
	createPair	External	✓	-
IDEXRouter	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IDividendDistributor	Interface			
	setDistributionCriteria	External	✓	-
	setShare	External	✓	-
	deposit	External	Payable	-
	process	External	✓	-
DividendDistributor	Implementation	IDividendDistributor		
		Public	✓	-
	setDistributionCriteria	External	✓	onlyToken
	setShare	External	✓	onlyToken
	deposit	External	Payable	onlyToken
	process	External	✓	onlyToken

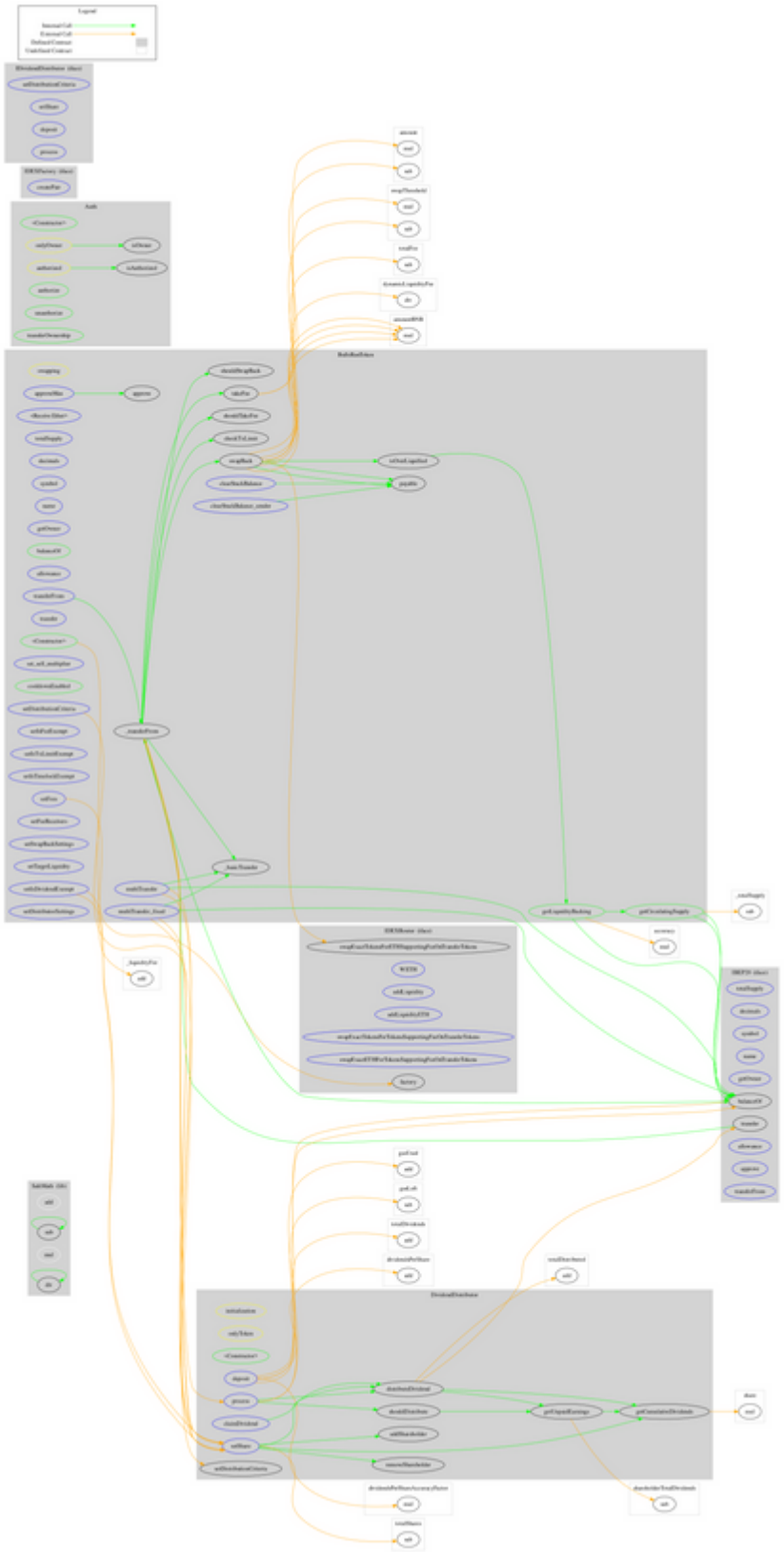
	shouldDistribute	Internal		
	distributeDividend	Internal	✓	
	claimDividend	External	✓	-
	getUnpaidEarnings	Public		-
	getCumulativeDividends	Internal		
	addShareholder	Internal	✓	
	removeShareholder	Internal	✓	
BullsRunToken	Implementation	IBEP20, Auth		
		Public	✓	Auth
		External	Payable	-
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	Public		-
	allowance	External		-
	approve	Public	✓	-
	approveMax	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	_transferFrom	Internal	✓	
	_basicTransfer	Internal	✓	
	checkTxLimit	Internal		
	shouldTakeFee	Internal		
	takeFee	Internal	✓	
	shouldSwapBack	Internal		
	clearStuckBalance	External	✓	authorized

	clearStuckBalance_sender	External	✓	authorized
	set_sell_multiplier	External	✓	onlyOwner
	cooldownEnabled	Public	✓	onlyOwner
	swapBack	Internal	✓	swapping
	setIsDividendExempt	External	✓	authorized
	setIsFeeExempt	External	✓	authorized
	setIsTxLimitExempt	External	✓	authorized
	setIsTimelockExempt	External	✓	authorized
	setFees	External	✓	authorized
	setFeeReceivers	External	✓	authorized
	setSwapBackSettings	External	✓	authorized
	setTargetLiquidity	External	✓	authorized
	setDistributionCriteria	External	✓	authorized
	setDistributorSettings	External	✓	authorized
	getCirculatingSupply	Public		-
	getLiquidityBacking	Public		-
	isOverLiquified	Public		-
	multiTransfer	External	✓	onlyOwner
	multiTransfer_fixed	External	✓	onlyOwner

Inheritance Graph



Flow Graph



Summary

Bulls Run 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 stopping transactions, transferring the user's tokens, and manipulating the fees. 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