Cyberscope

Audit Report
SPORT Token

March 2025

Network	BASE
Address	0xd007c4c900d1df6caea2a4122f3d551d7dfe08b0
Audited by	© cyberscope

Analysis

🔴 Critical 🟠 Medium ⚪ Minor / Informative 🔵 Pass

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

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	TSD	Total Supply Diversion	Unresolved
●	CCR	Contract Centralization Risk	Unresolved
●	IEE	Inconsistent Events Emission	Unresolved
●	MEE	Missing Events Emission	Unresolved
●	MU	Modifiers Usage	Unresolved
●	NWES	Nonconformity with ERC-20 Standard	Unresolved
●	PLPI	Potential Liquidity Provision Inadequacy	Unresolved
●	PSU	Potential Subtraction Underflow	Unresolved
●	RRA	Redundant Repeated Approvals	Unresolved

Table of Contents

Risk Classification

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

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

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

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

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

Review

Contract Name	SportToken
Compiler Version	v0.8.26+commit.8a97fa7a
Optimization	200 runs
Explorer	https://basescan.org/address/0xd007c4c900d1df6caea2a4122f3d551d7dfe08b0
Address	0xd007c4c900d1df6caea2a4122f3d551d7dfe08b0
Network	BASE
Symbol	SPORT
Decimals	18
Total Supply	105,000,000
Badge Eligibility	Yes

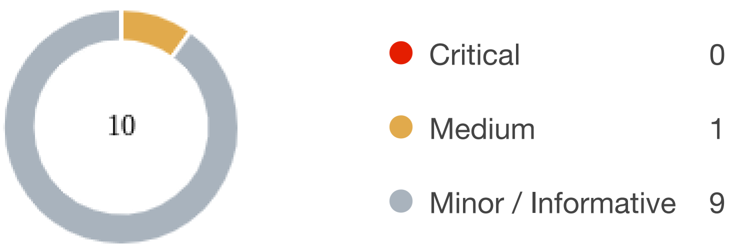
Audit Updates

Initial Audit	27 Feb 2025

Source Files

Filename	SHA256
SportToken.sol	70fb215a110969abab04b05dc60581e2acc7018c5bbbe5f97367271010e6c1ad

Findings Breakdown



Severity		Unresolved	Acknowledged	Resolved	Other
●	Critical	0	0	0	0
●	Medium	1	0	0	0
●	Minor / Informative	9	0	0	0

TSD - Total Supply Diversion

Criticality	Medium
Location	SportToken.sol#L1260,1281,1300,1320
Status	Unresolved

Description

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

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

This issue originates in the `transfer` method, which calculates reflection amounts using the ratios of `rSupply` and `tSupply` to update user balances. Initially, the method updates the balances of a subset of the involved accounts based on the existing ratio before invoking the `_reflectFee` method. When `_reflectFee` is executed, the reflection fee is deducted from `rSupply`, which alters the ratio of `rSupply` to `tSupply`. As a result, the subsequent state updates in the `transfer` method are performed using this new ratio. This inconsistency leads to a divergence between the cumulative balances held by user accounts and the total supply.

```
function _transferStandard(address sender, address recipient, uint256
tAmount) private {
(uint256 tTransferAmount, uint256 tFee, uint256 tLiquidity, uint256
tCoinOperation, uint256 tPrizeFund, uint256 tBurn) = _getValues(tAmount);
(uint256 rAmount, uint256 rTransferAmount, uint256 rFee) =
_getRValues(tAmount, tFee, tLiquidity, tCoinOperation,tPrizeFund, tBurn);
_rOwned[sender] = _rOwned[sender]-rAmount;
_rOwned[recipient] = _rOwned[recipient]+rTransferAmount;
_takeLiquidity(tLiquidity);
_reflectFee(rFee, tFee, tBurn);
_takeFund(tCoinOperation, operationFundWallet);
_takeFund(tPrizeFund, prizeFundWallet);
emit Transfer(sender, recipient, tTransferAmount);
}
```

Recommendation

The total supply and balance variables are separate and independent from each other. The total supply represents the total number of tokens that have been created, while the balance mapping stores the number of tokens owned by each account. Ideally, the sum of all balances should always equal the total supply. To achieve this, it is recommended that the team ensures a constant reflection rate is applied throughout the transfer process by executing the reflection of the fee after all relevant state updates have been completed.

ELFM - Exceeds Fees Limit

Criticality	Minor / Informative
Location	SportToken.sol#L1026
Status	Unresolved

Description

The contract owner has the authority to increase fees up to an allowed limit of 30%. The owner may take advantage of it by calling the `updateBuyTax` and the `updateSellTax` functions with a high percentage value.

```
function updateBuyTax(uint256 reflectionPer,uint256 coinOperartionPer,uint256
prizeFundPer,uint256 liquidityPer,uint256 burnPer) external onlyOwner {
require(reflectionPer <= 6,"You can not set reflection tax more then 6%");
require(coinOperartionPer <= 6,"You can not set coin operation  tax more then
6%");
require(prizeFundPer <= 6,"You can not set prizeFund tax more then 6%");
require(liquidityPer <= 6,"You can not set liquidity tax more then 6%");
require(burnPer <= 6,"You can not set burn  tax more then 6%");
buyReflectionTax= reflectionPer;
buyOperationWalletTax = coinOperartionPer;
buyPrizeFundWalletTax = prizeFundPer;
buyLiquidityTax = liquidityPer;
buyBurnTax = burnPer;
emit
BuyTaxUpdated(buyReflectionTax,buyOperationWalletTax,buyPrizeFundWalletTax,buy
LiquidityTax,buyBurnTax);
}
```

```
function updateSellTax(uint256 reflectionPer,uint256 coinOperartionPer,uint256
prizeFundPer,uint256 liquidityPer,uint256 burnPer) external onlyOwner {
require(reflectionPer <= 6,"You can not set reflection tax more then 6%");
require(coinOperartionPer <= 6,"You can not set coin operation  tax more then
6%");
require(prizeFundPer <= 6,"You can not set prizeFund tax more then 6%");
require(liquidityPer <= 6,"You can not set liquidity tax more then 6%");
require(burnPer <= 6,"You can not set burn  tax more then 6%");
sellReflectionTax= reflectionPer;
sellOperationWalletTax = coinOperartionPer;
sellPrizeFundWalletTax = prizeFundPer;
sellLiquidityTax = liquidityPer;
sellBurnTax = burnPer;
emit
BuyTaxUpdated(sellReflectionTax,sellOperationWalletTax,sellPrizeFundWalletTax,
sellLiquidityTax,sellBurnTax);
}
```

Recommendation

We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	SportToken.sol#L637,659,679,706,721,748,982,997,1020,1087
Status	Unresolved

Description

The contract's functionality and behavior are heavily dependent on external parameters or configurations. While external configuration can offer flexibility, it also poses several centralization risks that warrant attention. Centralization risks arising from the dependence on external configuration include Single Point of Control, Vulnerability to Attacks, Operational Delays, Trust Dependencies, and Decentralization Erosion.

```
function excludeFromReward(address account) public onlyOwner() {}
function includeInReward(address account) external onlyOwner() {}
function excludeFromFee(address account) external onlyOwner {}
function includeInFee(address account) external onlyOwner {}
function setOperationFundWallet(address wallet) external
onlyOwner{}
function setPrizeFundWallet(address wallet) external onlyOwner{}
function setSwapAndLiquifyEnabled(bool enabled) external onlyOwner
{}
function updateThreshold(uint256 amount) external onlyOwner {}
function startTrading() external onlyOwner(){}
function updateBuyTax(uint256 reflectionPer,uint256
coinOperartionPer,uint256 prizeFundPer,uint256 liquidityPer,uint256
burnPer) external onlyOwner {}
function updateSellTax(uint256 reflectionPer,uint256
coinOperartionPer,uint256 prizeFundPer,uint256 liquidityPer,uint256
burnPer) external onlyOwner {}
function airdrop(address[] calldata addresses, uint[] calldata

tokens) external onlyOwner {
```

Recommendation

To address this finding and mitigate centralization risks, it is recommended to evaluate the feasibility of migrating critical configurations and functionality into the contract's codebase itself. This approach would reduce external dependencies and enhance the contract's self-sufficiency. It is essential to carefully weigh the trade-offs between external configuration flexibility and the risks associated with centralization.

IEE - Inconsistent Events Emission

Criticality	Minor / Informative
Location	SportToken.sol#L1020
Status	Unresolved

Description

The contract executes actions and modifies its state through external methods, which lead to the emission of events. However, these events might be misleading or fail to accurately represent the actual state changes. Emitting inconsistent events can make it challenging for external parties to accurately assess the current state of the contract.

```
function updateSellTax(
uint256 reflectionPer,
uint256 coinOperartionPer,
uint256 prizeFundPer,
uint256 liquidityPer,
uint256 burnPer
) external onlyOwner {
...
emit BuyTaxUpdated(
sellReflectionTax,
sellOperationWalletTax,
sellPrizeFundWalletTax,
sellLiquidityTax,
sellBurnTax);
}
```

Recommendation

It is recommended to ensure that events emitted by the contract accurately reflect the actual state changes occurring within the contract. Significant actions should trigger corresponding events that include the relevant details.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	SportToken.sol#L790
Status	Unresolved

Description

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

```
function _takeFund(uint256 tFund, address fundWallet) private {
    uint256 currentRate = _getRate();
    uint256 rFund = tFund * currentRate;

    _rOwned[fundWallet] = _rOwned[fundWallet] + rFund;

    if (_isExcluded[fundWallet]) {
        _tOwned[fundWallet] = _tOwned[fundWallet] + tFund;
    }
}
```

Recommendation

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

MU - Modifiers Usage

Criticality	Minor / Informative
Location	SportToken.sol#L403,404,707,722,971,972,1042,1043
Status	Unresolved

Description

The contract is using repetitive statements on some methods to validate some preconditions. In Solidity, the form of preconditions is usually represented by the modifiers. Modifiers allow you to define a piece of code that can be reused across multiple functions within a contract. This can be particularly useful when you have several functions that require the same checks to be performed before executing the logic within the function.

```
require(wallet != address(0),"Fund wallet can not be zero");
```

Recommendation

The team is advised to use modifiers since it is a useful tool for reducing code duplication and improving the readability of smart contracts. By using modifiers to perform these checks, it reduces the amount of code that is needed to write, which can make the smart contract more efficient and easier to maintain.

NWES - Nonconformity with ERC-20 Standard

Criticality	Minor / Informative
Location	SportToken.sol#L1044
Status	Unresolved

Description

The contract does not fully conform to the ERC20 Standard. Specifically, according to the standard, transfers of 0 values must be treated as normal transfers and fire the Transfer event. However, the contract implements a conditional check that prohibits transfers of 0 values. This discrepancy between the contract's implementation and the ERC20 standard may lead to inconsistencies and incompatibilities with other contracts.

```
function _transfer( address from, address to, uint256 amount )
private{
require(from != address(0), "ERC20: transfer from the zero
address");
require(to != address(0), "ERC20: transfer to the zero address");
require(amount > 0, "Transfer amount must be greater than zero");
...
}
```

Recommendation

The incorrect implementation of the ERC20 standard could potentially lead to problems when interacting with the contract, as other contracts or applications that expect the ERC20 interface may not behave as expected. The team is advised to review and revise the implementation of the transfer mechanism to ensure full compliance with the ERC20 standard. https://eips.ethereum.org/EIPS/eip-20.

PLPI - Potential Liquidity Provision Inadequacy

Criticality	Minor / Informative
Location	SportToken.sol#L1176,1205
Status	Unresolved

Description

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

```
function swapTokensForEth(uint256 tokenAmount) private {
    // generate the uniswap pair path of token -> weth
    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = uniswapV2Router.WETH();

    _approve(address(this), address(uniswapV2Router), tokenAmount);

    // make the swap

uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
        tokenAmount,
        0, // accept any amount of ETH
        path,
        address(this),
        block.timestamp
    );
}
```

Recommendation

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

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

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

PSU - Potential Subtraction Underflow

Criticality	Minor / Informative
Location	SportToken.sol#L1232
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. In particular the contract does not ensure that the sender of the transaction holds the necessary amount of tokens before the transfer.

```
function _tokenTransfer(address sender, address recipient, uint256
amount,bool takeFee) private {
...
}
```

```
_tOwned[sender] = _tOwned[sender]-tAmount;
_rOwned[sender] = _rOwned[sender]-rAmount;
```

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.

RRA - Redundant Repeated Approvals

Criticality	Minor / Informative
Location	SportToken.sol#L1182,1207
Status	Unresolved

Description

The contract is designed to `approve` token transfers during the contract's operation by calling the _approve function before specific operations. This approach results in additional gas costs since the approval process is repeated for every operation execution, leading to inefficiencies and increased transaction expenses.

```
_approve(address(this), address(uniswapV2Router), tokenAmount);
    // make the swap

uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
        tokenAmount,
        0, // accept any amount of ETH
        path,
        address(this),
        block.timestamp
    );
```

Recommendation

Since the approved address is a trusted third-party source, it is recommended to optimize the contract by approving the maximum amount of tokens once in the initial set of the variable, rather than before each operation. This change will reduce the overall gas consumption and improve the efficiency of the contract.

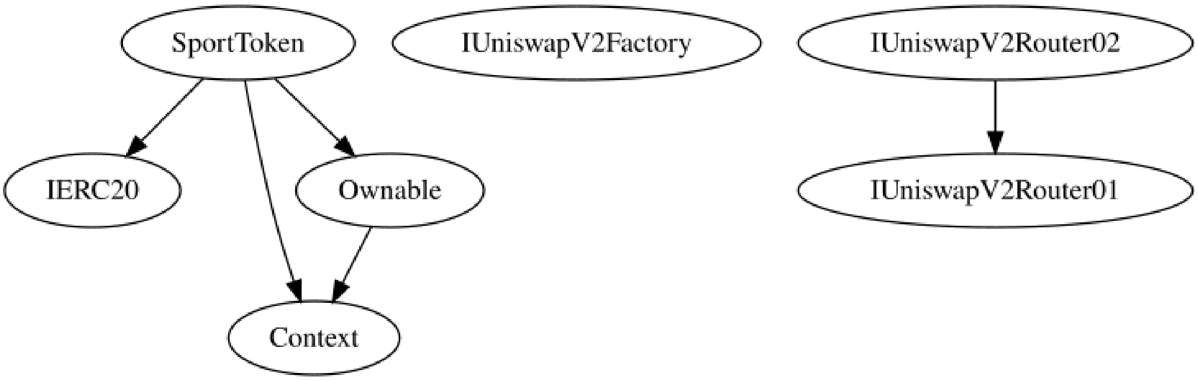
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
SportToken	Implementation	Context, IERC20, Ownable		
		Public	✓	-
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	Public		-
	transfer	External	✓	-
	allowance	External		-
	approve	Public	✓	-
	transferFrom	External	✓	-
	increaseAllowance	External	✓	-
	decreaseAllowance	External	✓	-
	isExcludedFromReward	External		-
	totalFees	External		-
	deliver	External	✓	-
	reflectionFromToken	External		-
	tokenFromReflection	Public		-
	excludeFromReward	Public	✓	onlyOwner

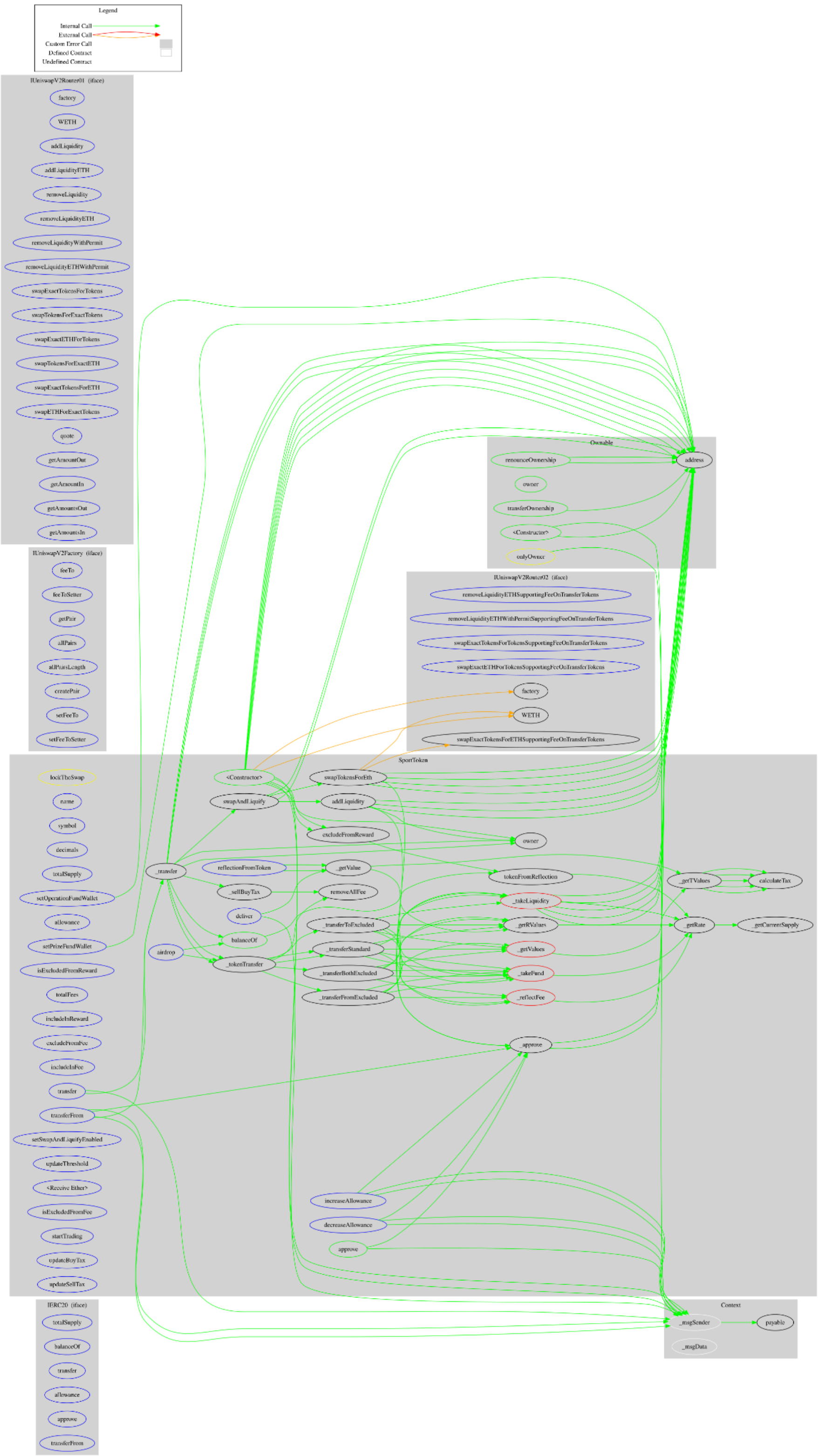
	includeInReward	External	✓	onlyOwner
	excludeFromFee	External	✓	onlyOwner
	includeInFee	External	✓	onlyOwner
	setOperationFundWallet	External	✓	onlyOwner
	setPrizeFundWallet	External	✓	onlyOwner
	setSwapAndLiquifyEnabled	External	✓	onlyOwner
	updateThreshold	External	✓	onlyOwner
		External	Payable	-
	_reflectFee	Private	✓	
	_takeFund	Private	✓	
	_getValues	Private		
	_getValue	Private		
	_getTValues	Private		
	_getRValues	Private		
	_getRate	Private		
	_getCurrentSupply	Private		
	_takeLiquidity	Private	✓	
	calculateTax	Private		
	removeAllFee	Private	✓	
	isExcludedFromFee	External		-
	_approve	Private	✓	
	startTrading	External	✓	onlyOwner
	updateBuyTax	External	✓	onlyOwner
	updateSellTax	External	✓	onlyOwner
	_transfer	Private	✓	

	airdrop	External	✓	onlyOwner
	_sellBuyTax	Private	✓	
	swapAndLiquify	Private	✓	lockTheSwap
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	_tokenTransfer	Private	✓	
	_transferBothExcluded	Private	✓	
	_transferStandard	Private	✓	
	_transferToExcluded	Private	✓	
	_transferFromExcluded	Private	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Custom Error Call
Defined Contract
Undefined Contract
IUniswapV2Router01 (iface)
factory
WETH
addLiquidity
addLiquidityETH
removeLiquidity
removeLiquidityETH
removeLiquidityWithPermit
removeLiquidityETHWithPermit
swapExactTokensForTokens
swapTokensForExactTokens
swapExactETHForTokens
swapTokensForExactETH
swapExactTokensForETH
swapETHForExactTokens
quote
getAmountOut
getAmountIn
getAmountsOut
getAmountsIn
IUniswapV2Factory (iface)
feeTo
feeToSetter
getPair
allPairs
allPairsLength
createPair
setFeeTo
setFeeToSetter
Ownable
renounceOwnership
address
owner
transferOwnership
<Constructor>
onlyOwner
IUniswapV2Router02 (iface)
removeLiquidityETHSupportingFeeOnTransferTokens
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens
factory
WETH
swapExactTokensForETHSupportingFeeOnTransferTokens
SportToken
lockTheSwap
<Constructor>
swapTokensForEth
name
swapAndLiquify
addLiquidity
symbol
excludeFromReward
decimals
owner
totalSupply
_transfer
reflectionFromToken
_getValue
tokenFromReflection
_getTValues
calculateTax
setOperationFundWallet
_sellBuyTax
removeAllFee
_takeLiquidity
allowance
deliver
_transferToExcluded
_getRValues
_getRate
_getCurrentSupply
setPrizeFundWallet
balanceOf
airdrop
_transferStandard
_getValues
_tokenTransfer
isExcludedFromReward
_transferBothExcluded
_takeFund
totalFees
_transferFromExcluded
_reflectFee
includeInReward
excludeFromFee
_approve
includeInFee
transfer
transferFrom
setSwapAndLiquifyEnabled
updateThreshold
<Receive Ether>
increaseAllowance
isExcludedFromFee
decreaseAllowance
startTrading
approve
updateBuyTax
updateSellTax
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender
payable
_msgData

Summary

SPORT 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 manipulate 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. There is also a limit of max 30% fees.

Disclaimer

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

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope’s position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

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

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

The Cyberscope team

cyberscope.io