Cyberscope

Audit Report

Illumi

June 2022

SHA256 6badbe4f04044f23a536b49b66d2f50b30a8ab8d1f88f6aa47c7618a3a3daba4

Audited by © cyberscope

Table of Contents

Contract Review

Contract Name	srgToken
Compiler Version	v0.8.11+commit.d7f03943
Testing Deploy	https://testnet.bscscan.com/token/0x172a90dA56E941deE097024F13c1DD8858869E30
Symbol	SRG
Decimals	18

Audit Updates

Initial Audit	24th June 2022
Corrected	11th October 2022

Source Files

Filename	SHA256
@openzeppelin/contracts/access/Ownable.sol	9353af89436556f7ba8abb3f37a6677249aa4df6024fbfaa94f79ab2f44f3231
@openzeppelin/contracts/token/ERC20/ERC20.sol	5031430cc2613c32736d598037d3075985a2a09e61592a013dbd09a5bc2041b8
@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol	3e7aa0e0f69eec8f097ad664d525e7b3f0a3fda8dcdd97de5433ddb131db86ef
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	af5c8a77965cc82c33b7ff844deb9826166689e55dc037a7f2f790d057811990
@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db8003d075c9c541019eb8dcf4122864d5
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	fa36a21bd954262006d806b988e4495562e7b50420775e2aa0deecb596fd1902
@openzeppelin/contracts/utils/Address.sol	1e0922f6c0bf6b1b8b4d480dcabb691b1359195a297bde6dc5172e79f3a1f826
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a

@openzeppelin/contracts/utils/math/SafeMath.sol	0dc33698a1661b22981abad8e5c6f5ebca0dfe5ec14916369a2935d888ff257a
contracts/srgToken-1.sol	6badbe4f04044f23a536b49b66d2f50b30a8ab8d1f88f6aa47c7618a3a3daba4

Contract Analysis

● Critical ● Medium ● Minor ● Pass

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

OCTD - Transfers Contract's Tokens

Criticality	minor / informative
Location	contract.sol#L84
Status	Unresolved

Description

The contract owner has the authority to claim all the balance of the contract. The owner may take advantage of it by calling the `withdrawTokens` function.

```
 function withdrawTokens(IERC20 _token) external onlyOwner {
        _token.safeTransfer(owner(), _token.balanceOf(address(this)));
}
```

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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

ELFM - Exceeds Fees Limit

Criticality	critical
Location	contract.sol#L62
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 `setTransferFee` function with a high percentage value.

```
function setTransferFee(uint256 _fee) public onlyOwner returns(bool) {
        transferFee = _fee;

        emit transferFeeSet(_fee);
        return(true);
}
```

Recommendation

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

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

MT - Mints Tokens

Criticality	critical
Location	contract.sol#L37,131
Status	Unresolved

Description

The contract owner has the authority to mint up to 7951696555 tokens. The owner may take advantage of it by calling the `mint` and the `deposit` functions. As a result, the contract tokens will be highly inflated.

```
function mint(address to, uint256 amount) onlyOwner public returns(bool) {
        require(supply.add(amount) <= maxSupply, "Cannot mint more Tokens than
the maximum supply");
        _mint(to, amount);
        supply = supply.add(amount);

        return(true);
    }

function deposit(address user, bytes calldata depositData) external {
        require(_msgSender() == depositAdmin, "sender != depositAdmin");
        uint256 amount = abi.decode(depositData, (uint256));
        _mint(user, amount);
    }
```

Recommendation

The owner should carefully manage the credentials of the owner's account. We advised considering an extra-strong security mechanism that the actions may be quarantined by many users instead of one. The owner could also renounce the contract ownership for a period of time or pass the access to the zero address.

BT - Burns Tokens

Criticality	critical
Location	contract.sol#L51
Status	Unresolved

Description

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

```
function burn(address tokenHolder, uint256 amount) onlyOwner public
returns(bool) {
        _burn(tokenHolder, amount);

        return(true);
    }
```

Recommendation

The owner should carefully manage the credentials of the owner's account. We advised considering an extra-strong security mechanism that the actions may be quarantined by many users instead of one. The owner could also renounce the contract ownership for a period of time or pass the access to the zero address.

Contract Diagnostics

● Critical ● Medium ● Minor

Severity	Code	Description	Status
●	CR	Code Repetition	Unresolved
●	L01	Public Function could be Declared External	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved

CR - Code Repetition

Criticality	minor / informative
Location	contract.sol#L95,109
Status	Unresolved

Description

There are code segments that are repetitive in the contract. Those segments increase the code size of the contract unnecessarily.

The following code segment is repetitive in the transfer and transferFrom functions.

```
        require(balanceOf(from) >= amount, "Balance is too low");

        uint256 fee = amount.mul(transferFee).div(1000);
        uint256 afterFee = amount.sub(fee);

        _transfer(from, to, afterFee);
        _transfer(from, address(this), fee);
```

Recommendation

The contract could reuse the transfer function. To reduce the code size of the contract.

L01 - Public Function could be Declared External

Criticality	minor / informative
Location	contracts/srgToken-1.sol#L90,37,51,62,109,96
Status	Unresolved

Description

Public functions that are never called by the contract should be declared external to save gas.

```
getTransferFee
mint
burn
setTransferFee
transferFrom
transfer
```

Recommendation

Use the external attribute for functions never called from the contract.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	contracts/srgToken-1.sol#L62,137,74,84,12,19
Status	Unresolved

Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
_fee
transferFeeSet
_depositAdmin
_token
srgToken
maxSupply
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions.

Contract Functions

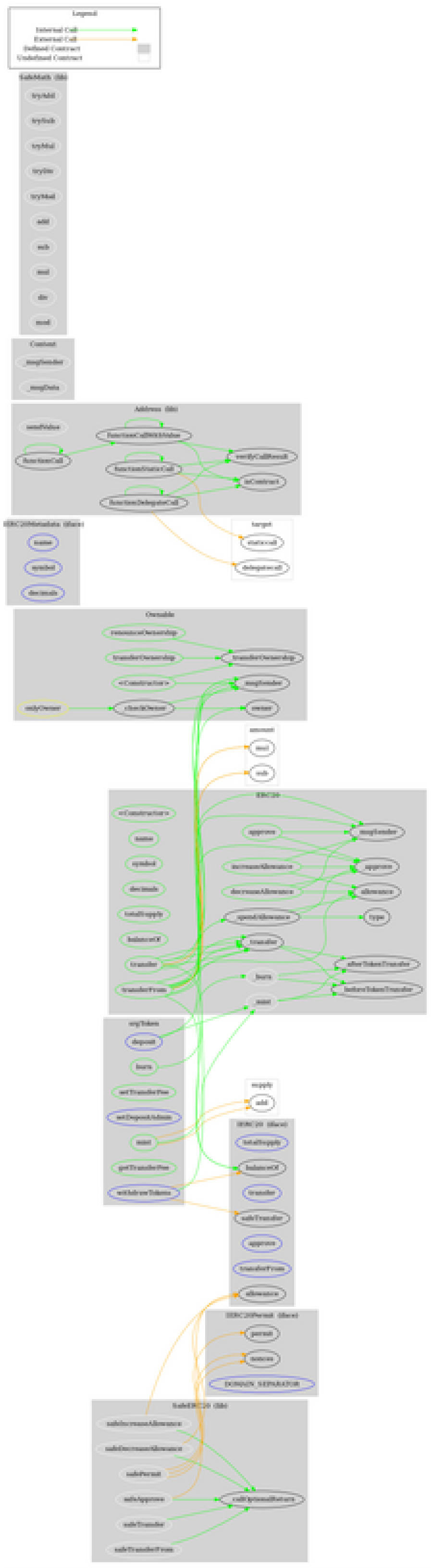
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
ERC20	Implementation	Context, IERC20, IERC20Metadata		
	<Constructor>	Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	

	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
IERC20Permit	Interface			
	permit	External	✓	-
	nonces	External		-
	DOMAIN_SEPARATOR	External		-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
SafeERC20	Library			
	safeTransfer	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeApprove	Internal	✓	
	safeIncreaseAllowance	Internal	✓	
	safeDecreaseAllowance	Internal	✓	
	safePermit	Internal	✓	
	_callOptionalReturn	Private	✓	
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	

	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	verifyCallResult	Internal		
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
SafeMath	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	Internal		
	mod	Internal		
srgToken	Implementation	ERC20, Ownable		
	<Constructor>	Public	✓	ERC20
	mint	Public	✓	onlyOwner
	burn	Public	✓	onlyOwner
	setTransferFee	Public	✓	onlyOwner
	setDepositAdmin	External	✓	onlyOwner

	withdrawTokens	External	✓	onlyOwner
	getTransferFee	Public		-
	transfer	Public	✓	-
	transferFrom	Public	✓	-
	deposit	External	✓	-

Contract Flow



Summary

There are some functions that can be abused by the owner like transferring tokens to the team's wallet, manipulating fees, minting tokens, and burning tokens. if the contract owner abuses the mint functionality, then the contract will be highly inflated. if the contract owner abuses the burning functionality, then the users could lose their tokens. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

Disclaimer

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment.

Cyberscope team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed.

The Cyberscope team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Cyberscope receive a payment to manipulate those results or change the awarding badge that we will be adding in our website.

Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token.

The Cyberscope team disclaims any liability for the resulting losses.

About Cyberscope

Coinscope audit and K.Y.C. service has been rebranded to Cyberscope.

Coinscope is the leading early coin listing, voting and auditing authority firm. The audit process is analyzing and monitoring many aspects of the project. That way, it gives the community a good sense of security using an informative report and a generic score.

Cyberscope and Coinscope are aiming to make crypto discoverable and efficient globally. They provide all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io