



Cyberscope

Audit Report

SciMatic+ COIN

March 2023

Type BEP20

Network BSC

Address 0x04524f05CCE069aF9BcE7879616773A94bb46C84

Audited by © cyberscope

Table of Contents

Table of Contents	1
Review	2
Audit Updates	2
Source Files	2
Analysis	3
ST - Stops Transactions	4
Description	4
Recommendation	4
MT - Mints Tokens	5
Description	5
Recommendation	5
BC - Blacklists Addresses	6
Description	6
Recommendation	6
Diagnostics	7
L04 - Conformance to Solidity Naming Conventions	8
Description	8
Recommendation	8
L11 - Unnecessary Boolean equality	9
Description	9
Recommendation	9
L19 - Stable Compiler Version	10
Description	10
Recommendation	10
Functions Analysis	11
Inheritance Graph	15
Flow Graph	16
Summary	17
Disclaimer	18
About Cyberscope	19

Review

Contract Name	SCIp
Compiler Version	v0.8.9+commit.e5eed63a
Optimization	200 runs
Explorer	https://bscscan.com/address/0x04524f05cce069af9bce7879616773a94bb46c84
Address	0x04524f05cce069af9bce7879616773a94bb46c84
Network	BSC
Symbol	SCI+
Decimals	18
Total Supply	10.000.000.000

Audit Updates

Initial Audit	13 Mar 2023
----------------------	-------------

Source Files

Filename	SHA256
SCIp.sol	7de80a91051b7c2e823a7a21788bdbf39614c3256588fafc650d0fd44628c6c7

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	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Unresolved
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Unresolved

ST - Stops Transactions

Criticality	Minor / Informative
Location	SClp.sol#L955
Status	Unresolved

Description

The contract owner has the authority to stop the transactions for all users. The owner may take advantage of it by calling the `pause` function.

```
function _beforeTokenTransfer(address from, address to, uint256 amount)
    internal
    whenNotPaused
    override
{
    super._beforeTokenTransfer(from, to, amount);
}
```

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.

MT - Mints Tokens

Criticality	Critical
Location	SClp.sol#L946
Status	Unresolved

Description

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

```
function mint(address to, uint256 amount) public onlyOwner {  
    _mint(to, amount);  
}
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

BC - Blacklists Addresses

Criticality	Medium
Location	SClp.sol#L445
Status	Unresolved

Description

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

```
function blockAddress(address _address) public returns (bool) {  
    require(isAdmin(), "You are not admin");  
    _blocked[_address] = true;  
    return true;  
}
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L11	Unnecessary Boolean equality	Unresolved
●	L19	Stable Compiler Version	Unresolved

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	SCIp.sol#L445,454
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
address _address
```

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>.

L11 - Unnecessary Boolean equality

Criticality	Minor / Informative
Location	SClp.sol#L508,533,561,582,604
Status	Unresolved

Description

Boolean equality is unnecessary when comparing two boolean values. This is because a boolean value is either true or false, and there is no need to compare two values that are already known to be either true or false.

it's important to be aware of the types of variables and expressions that are being used in the contract's code, as this can affect the contract's behavior and performance. The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
require(isBlocked() != true, "blocked")
```

Recommendation

Using the boolean value itself is clearer and more concise, and it is generally considered good practice to avoid unnecessary boolean equalities in Solidity code.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	SClp.sol#L3
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.9;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

Functions Analysis

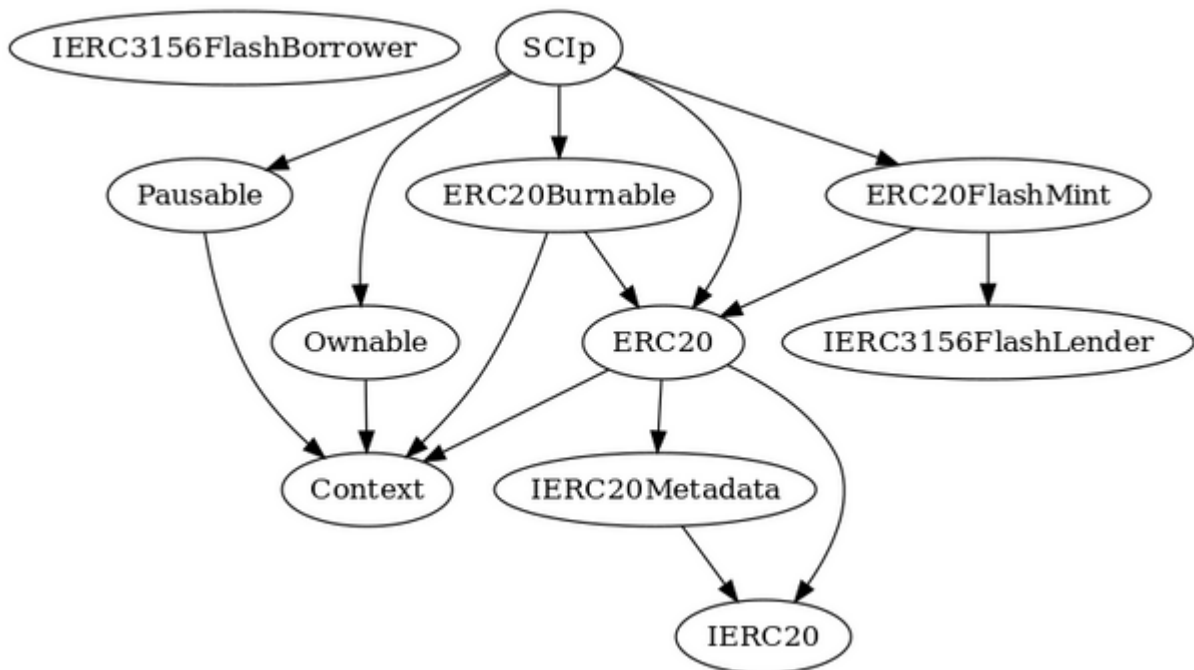
Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
IERC3156FlashBorrower	Interface			
	onFlashLoan	External	✓	-
IERC3156FlashLender	Interface			
	maxFlashLoan	External		-
	flashFee	External		-
	flashLoan	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
Pausable	Implementation	Context		
		Public	✓	-
	paused	Public		-

	_requireNotPaused	Internal		
	_requirePaused	Internal		
	_pause	Internal	✓	whenNotPaused
	_unpause	Internal	✓	whenPaused
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
ERC20	Implementation	Context, IERC20, IERC20Metadata		
		Public	✓	-
	name	Public		-
	isAdmin	Public		-
	isBlocked	Public		-
	blockAddress	Public	✓	-
	unBlockAddress	Public	✓	-
	symbol	Public		-
	decimals	Public		-

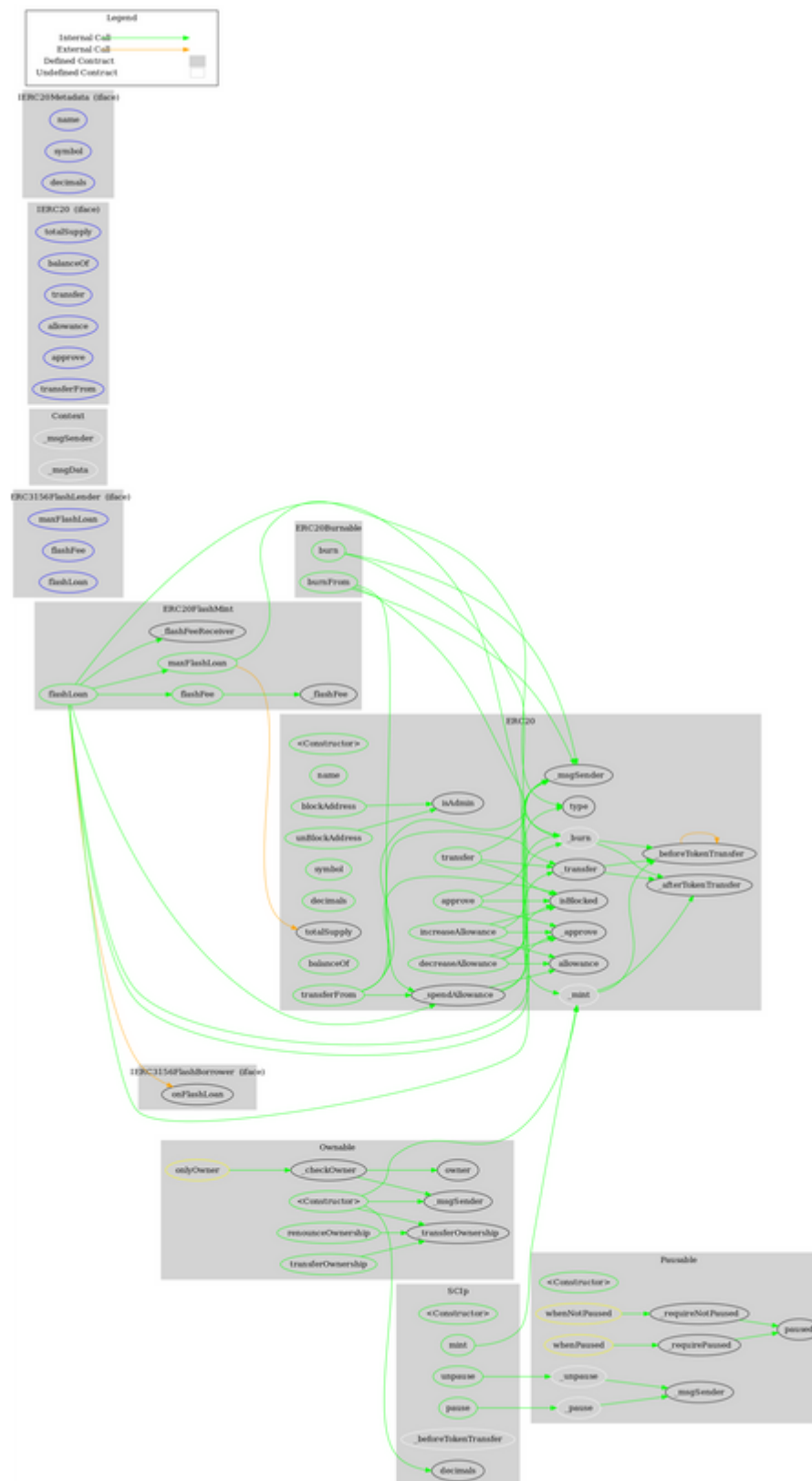
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
ERC20FlashMint	Implementation	ERC20, IERC3156FlashLender		
	maxFlashLoan	Public		-
	flashFee	Public		-
	_flashFee	Internal		
	_flashFeeReceiver	Internal		
	flashLoan	Public	✓	-
ERC20Burnable	Implementation	Context, ERC20		
	burn	Public	✓	-
	burnFrom	Public	✓	-

SCIp	Implementation	ERC20, ERC20Burn able, Pausable, Ownable, ERC20Flash Mint		
		Public	✓	ERC20
	pause	Public	✓	onlyOwner
	unpause	Public	✓	onlyOwner
	mint	Public	✓	onlyOwner
	_beforeTokenTransfer	Internal	✓	whenNotPaus ed

Inheritance Graph



Flow Graph



Summary

SciMatic+ COIN contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. There are some functions that can be abused by the owner like stop transactions, mint tokens and blacklist addresses. If the contract owner abuses the mint functionality, then the contract will be highly inflated. 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>