



Cyberscope

Audit Report

SecureChain AI

September 2023

Network ETH,BSC

Address (ETH) 0xE35009059cb55ded065027e9832A2c564AFF7512

Address (BSC) 0x051A66a7750098fB1EC6548D36E275bb23749A78

Audited by © cyberscope

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

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

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	RSW	Redundant Storage Writes	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L09	Dead Code Elimination	Unresolved

Table of Contents

Analysis	1
Diagnostics	2
Table of Contents	3
Review	4
Audit Updates	4
Source Files	5
Findings Breakdown	6
ST - Stops Transactions	7
Description	7
Recommendation	7
RSW - Redundant Storage Writes	8
Description	8
Recommendation	8
L02 - State Variables could be Declared Constant	9
Description	9
Recommendation	9
IDI - Immutable Declaration Improvement	10
Description	10
Recommendation	10
L04 - Conformance to Solidity Naming Conventions	11
Description	11
Recommendation	12
L09 - Dead Code Elimination	13
Description	13
Recommendation	13
Functions Analysis	14
Inheritance Graph	18
Flow Graph	19
Summary	20
Disclaimer	21
About Cyberscope	22

Review

Contract Name	SCAI
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	200 runs
Explorer (ETH)	https://etherscan.io/address/0xe35009059cb55ded065027e9832a2c564aff7512
Explorer (BSC)	https://bscscan.com/address/0x051a66a7750098fb1ec6548d36e275bb23749a78
Address (ETH)	0xe35009059cb55ded065027e9832a2c564aff7512
Address (BSC)	0x051a66a7750098fb1ec6548d36e275bb23749a78
Network	ETH,BSC
Symbol	SCAI
Decimals	18
Total Supply	100,000,000

Audit Updates

Initial Audit	31 Aug 2023
---------------	-------------

Source Files

Filename	SHA256
SCAI.sol (ETH)	18aacd0ee1bdd76635b1dd95057857db3a78f5254618715a3a0dcb779946947b
SCAI.sol (BSC)	1269e9f3bd129da2b84c69b3eef3e47a4c97d2e0aff702b85a03511d6e9a1458

Findings Breakdown



Critical	1
Medium	0
Minor / Informative	5

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	1	0	0	0
Medium	0	0	0	0
Minor / Informative	5	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	SCAI.sol#L306
Status	Unresolved

Description

The transactions are initially disabled for all users excluding the authorized addresses. The owner can enable the transactions for all users. Once the transactions are enabled the owner will not be able to disable them again.

```
if (isLimitedAddress(from,to)) {  
    require(isTradingEnabled,"Trading is not enabled");  
}
```

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 multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	SCAI.sol#L257
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract modifies the state of certain variables even when their current state is equal to the provided argument. As a result, the contract performs redundant storage writes.

```
_noFee[account] = enabled
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	SCAI.sol#L169,170,171
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.

```
uint256 public buyfee = 20  
uint256 public sellfee = 20  
uint256 public transferfee = 0
```

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.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	SCAI.sol#L204,220
Status	Unresolved

Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
swapRouter  
lpPair
```

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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	SCAI.sol#L69,178,179,180,193,194,195,196,197,268,273,278,389
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);
string constant private _name = "SecureChain AI"
string constant private _symbol = "SCAI"
uint8 constant private _decimals = 18
event _enableTrading();
event _setPresaleAddress(address account, bool enabled);
event _toggleCanSwapFees(bool enabled);
event _changePair(address newLpPair);
event _changeWallets(address marketing);

function is_buy(address ins, address out) internal view returns (bool) {
    bool _is_buy = !isLpPair[out] && isLpPair[ins];
    return _is_buy;
}

...
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation

<https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention>.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	SCAI.sol#L278
Status	Unresolved

Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function is_transfer(address ins, address out) internal view returns
(bool) {
    bool _is_transfer = !isLpPair[out] && !isLpPair[ins];
    return _is_transfer;
}
```

Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

Functions Analysis

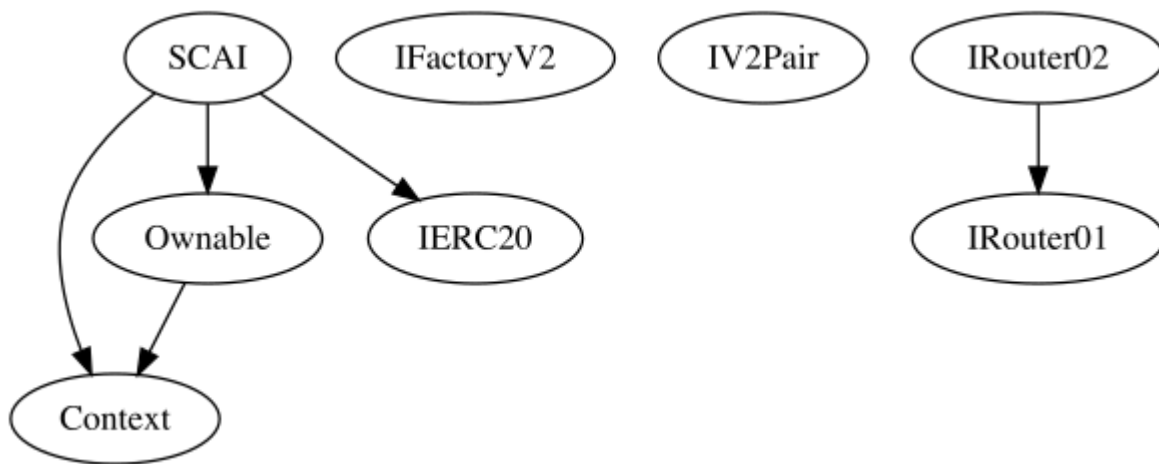
Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
Context	Implementation			
		Public	✓	-
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_setOwner	Private	✓	
IFactoryV2	Interface			
	getPair	External		-
	createPair	External	✓	-
IV2Pair	Interface			
	factory	External		-

	getReserves	External		-
	sync	External	✓	-
IRouter01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
	addLiquidity	External	✓	-
	swapExactETHForTokens	External	Payable	-
	getAmountsOut	External		-
	getAmountsIn	External		-
IRouter02	Interface	IRouter01		
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokens	External	✓	-
IERC20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-

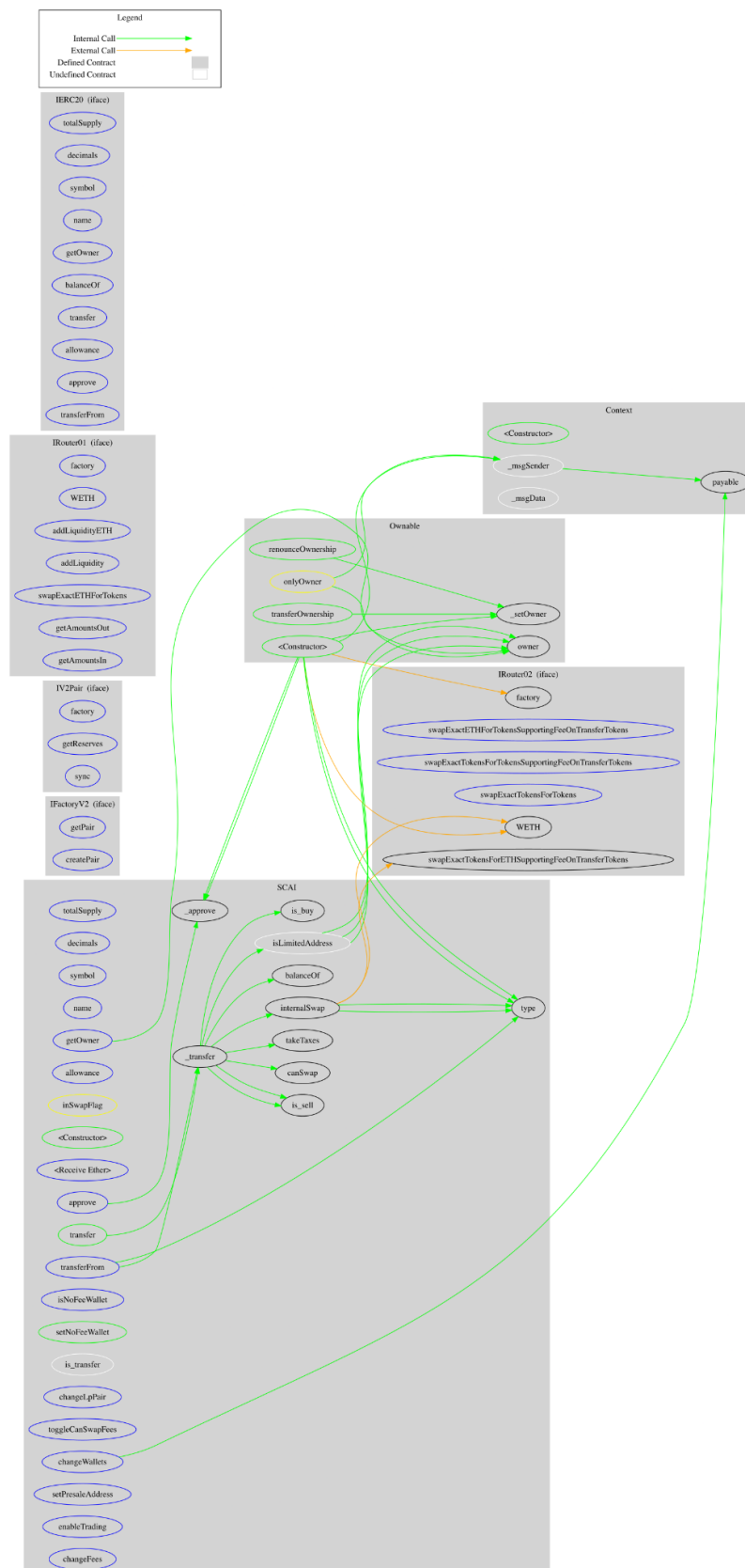
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
SCAI	Implementation	Context, Ownable, IERC20		
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	allowance	External		-
	balanceOf	Public		-
		Public	✓	-
		External	Payable	-
	transfer	Public	✓	-
	approve	External	✓	-
	_approve	Internal	✓	
	transferFrom	External	✓	-
	isNoFeeWallet	External		-
	setNoFeeWallet	Public	✓	onlyOwner

	isLimitedAddress	Internal		
	is_buy	Internal		
	is_sell	Internal		
	is_transfer	Internal		
	canSwap	Internal		
	changeLpPair	External	✓	onlyOwner
	toggleCanSwapFees	External	✓	onlyOwner
	_transfer	Internal	✓	
	changeWallets	External	✓	onlyOwner
	takeTaxes	Internal	✓	
	internalSwap	Internal	✓	inSwapFlag
	setPresaleAddress	External	✓	onlyOwner
	enableTrading	External	✓	onlyOwner
	changeFees	External	✓	onlyOwner

Inheritance Graph



Flow Graph



Summary

SecureChain AI 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. A multi-wallet signing pattern will provide security against potential hacks. There is also a limit of max 2% fees on BSC network. The fees are fixed on the ETH network at 2% buy/sell fees and 0% transfer 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

<https://www.cyberscope.io>