



# Cyberscope

## Audit Report

# ISOMA

Aug 2023

Commit 918bb306f6b66c725375a128ac88bce056e31159

Repository [https://github.com/isoma-project/isoma\\_blockchain/blob/master/smart\\_contract/ISOMATest.sol](https://github.com/isoma-project/isoma_blockchain/blob/master/smart_contract/ISOMATest.sol)

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
●	ZD	Zero Division	Unresolved
●	PVC	Price Volatility Concern	Unresolved
●	RSW	Redundant Storage Writes	Unresolved
●	MEE	Missing Events Emission	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
●	L09	Dead Code Elimination	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

# Table of Contents

<b>Analysis</b>	<b>1</b>
<b>Diagnostics</b>	<b>2</b>
<b>Table of Contents</b>	<b>3</b>
<b>Review</b>	<b>5</b>
Audit Updates	5
Source Files	5
<b>Findings Breakdown</b>	<b>6</b>
ST - Stops Transactions	7
Description	7
Recommendation	7
ZD - Zero Division	8
Description	8
Recommendation	8
PVC - Price Volatility Concern	10
Description	10
Recommendation	10
RSW - Redundant Storage Writes	11
Description	11
Recommendation	11
MEE - Missing Events Emission	12
Description	12
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	16
L07 - Missing Events Arithmetic	18
Description	18
Recommendation	18
L09 - Dead Code Elimination	19
Description	19
Recommendation	19
L13 - Divide before Multiply Operation	21
Description	21

Recommendation	21
L20 - Succeeded Transfer Check	22
Description	22
Recommendation	22
<b>Functions Analysis</b>	<b>23</b>
<b>Inheritance Graph</b>	<b>29</b>
<b>Flow Graph</b>	<b>30</b>
<b>Summary</b>	<b>31</b>
<b>Disclaimer</b>	<b>32</b>
<b>About Cyberscope</b>	<b>33</b>

## Review

Repository	<a href="https://github.com/isoma-project/isoma_blockchain/blob/master/smart_contract/ISOMATest.sol">https://github.com/isoma-project/isoma_blockchain/blob/master/smart_contract/ISOMATest.sol</a>
Commit	918bb306f6b66c725375a128ac88bce056e31159

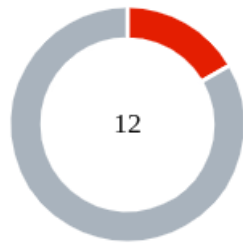
## Audit Updates

Initial Audit	03 Aug 2023
---------------	-------------

## Source Files

Filename	SHA256
ISOMATest.sol	265769150eafa708fd1ed647dfe13feff476fe57407d9ffafd4f289d3f7adaf

## Findings Breakdown



Critical	2
Medium	0
Minor / Informative	10

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	2	0	0	0
Medium	0	0	0	0
Minor / Informative	10	0	0	0

## ST - Stops Transactions

Criticality	Critical
Location	ISOMATest.sol#L762
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 enable the owner will not be able to disable them again.

```
if (!tradingActive) {  
    require(  
        _isExcludedFromFees[from] || _isExcludedFromFees[to],  
        "Trading is not enabled yet."  
    );  
}
```

### 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.



## ZD - Zero Division

Criticality	Critical
Location	ISOMATest.sol#L878
Status	Unresolved

### Description

The contract is using variables that may be set to zero as denominators. This can lead to unpredictable and potentially harmful results, such as a transaction revert.

Specifically, the contract calculates the `lpTokens` value using the `totalFee` variable as a denominator. However, if all fee values are set to zero, then both `totalBuyFee` and `totalSellFee` would be zero, resulting in `totalFee` also being zero. This would cause the `lpTokens` calculation to attempt a division by zero, leading to a transaction revert.

```
function swapBack() private {  
    ...  
    if (contractBalance >= swapTokensAtAmount) {  
        uint256 totalBuyFee = marketingAndDevFeeBuy +  
charityFeeBuy + autoLPFeeBuy;  
        uint256 totalSellFee = marketingAndDevFeeSell +  
charityFeeSell + autoLPFeeSell;  
        uint256 totalFee = totalBuyFee + totalSellFee;  
        uint256 lpFee = (autoLPFeeBuy + autoLPFeeSell) / 2;  
        uint256 lpTokens = (contractBalance * (autoLPFeeBuy  
+ autoLPFeeSell)) / totalFee;  
        ...  
    }  
}
```

### Recommendation

It is important to handle division by zero appropriately in the code to avoid unintended behavior and to ensure the reliability and safety of the contract. The contract should ensure that the divisor is always non-zero before performing a division operation. It should prevent the variables to be set to zero, or should not allow the execution of the corresponding

statements. Specifically, the team is advised to add a condition to check if `totalFee` is zero before performing the division.

## PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	ISOMATest.sol#L700
Status	Unresolved

### Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapTokensAtAmount` 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 updateSwapTokensAmount(uint256 newAmount) external  
onlyOwner {  
    swapTokensAtAmount = newAmount;  
}
```

### 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.

## RSW - Redundant Storage Writes

<b>Criticality</b>	Minor / Informative
<b>Location</b>	ISOMATest.sol#L692
<b>Status</b>	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 updates the state of excluded addresses even if their current state is the same as the the one passed as an argument. As a result, the contract performs redundant storage writes.

```
function excludeFromFees(address account, bool excluded)
public onlyOwner {
    _isExcludedFromFees[account] = excluded;
}
```

### 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.

## MEE - Missing Events Emission

Criticality	Minor / Informative
Location	ISOMATest.sol#L643,657
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.

Specifically, the functions `updateBuyFee` , `updateSellFee` and `excludeFromFees` are missing event emissions.

```
function updateBuyFee(uint16 marketing, uint16 charity, uint16
autoLP, uint16 staking) external onlyOwner {
    require(marketing + charity + autoLP + staking <= 10,
"Max buy fees limit is 10 percent");
    marketingAndDevFeeBuy = marketing;
    charityFeeBuy = charity;
    autoLPFeeBuy = autoLP;
    stakingFeeBuy = staking;
}

function updateSellFee(uint16 marketing, uint16 charity,
uint16 autoLP, uint16 staking) external onlyOwner {
    ...

function excludeFromFees(address account, bool excluded)
public onlyOwner {
    _isExcludedFromFees[account] = excluded;
}
```

## 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.

## IDI - Immutable Declaration Improvement

<b>Criticality</b>	Minor / Informative
<b>Location</b>	ISOMATest.sol#L605,606
<b>Status</b>	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.

```
uniswapV2Router
uniswapV2Pair
```

### 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	ISOMATest.sol#L568
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 _totalSupply = 100_000_000_000_000_000 * 1e18
```

### 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	ISOMATest.sol#L431,433,464,506,568,596
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 DOMAIN_SEPARATOR() external view returns (bytes32);
function PERMIT_TYPEHASH() external pure returns (bytes32);
function MINIMUM_LIQUIDITY() external pure returns (uint256);
function WETH() external pure returns (address);
uint256 _totalSupply = 100_000_000_000_000_000 * 1e18
mapping(address => bool) public _isExcludedMaxTransactionAmount
```

### 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

<b>Criticality</b>	Minor / Informative
<b>Location</b>	ISOMATest.sol#L645,659,700
<b>Status</b>	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.

```
marketingAndDevFeeBuy = marketing
marketingAndDevFeeSell = marketing
swapTokensAtAmount = newAmount
```

### 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.

## L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	ISOMATest.sol#L324
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 _burn(address account, uint256 amount) internal
virtual {
    require(account != address(0), "ERC20: burn from the
zero address");

    _beforeTokenTransfer(account, address(0), amount);

    uint256 accountBalance = _balances[account];
    ...
}
_totalSupply -= amount;

emit Transfer(account, address(0), amount);

_afterTokenTransfer(account, address(0), amount);
}
```

### 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.

## L13 - Divide before Multiply Operation

<b>Criticality</b>	Minor / Informative
<b>Location</b>	ISOMATest.sol#L829,830,835,836
<b>Status</b>	Unresolved

### Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of precision.

```
fees = (amount * sellFees) / 100
stakingFee = (fees * stakingFeeSell) / sellFees
```

### Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

## L20 - Succeeded Transfer Check

<b>Criticality</b>	Minor / Informative
<b>Location</b>	ISOMATest.sol#L686
<b>Status</b>	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.

```
token.transfer(owner(), balance)
```

### 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
<b>Context</b>	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
<b>Ownable</b>	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
<b>IERC20</b>	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-



<b>IERC20Metadata</b>	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
<b>ERC20</b>	Implementation	Context, IERC20, IERC20Meta data		
		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	✓	

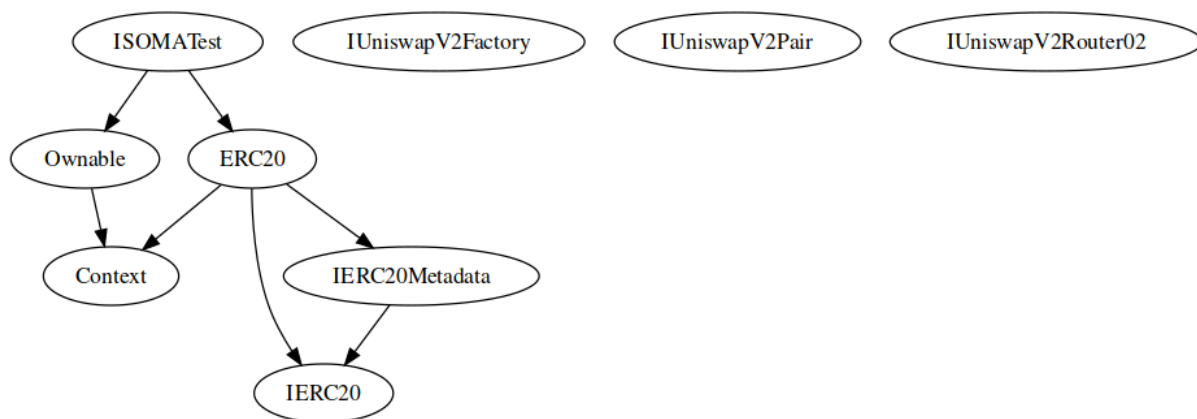
	_approve	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
<b>IUniswapV2Factory</b>	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
<b>IUniswapV2Pair</b>	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-

	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	mint	External	✓	-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-
	initialize	External	✓	-
<b>IUniswapV2Router02</b>	Interface			
	factory	External		-
	WETH	External		-

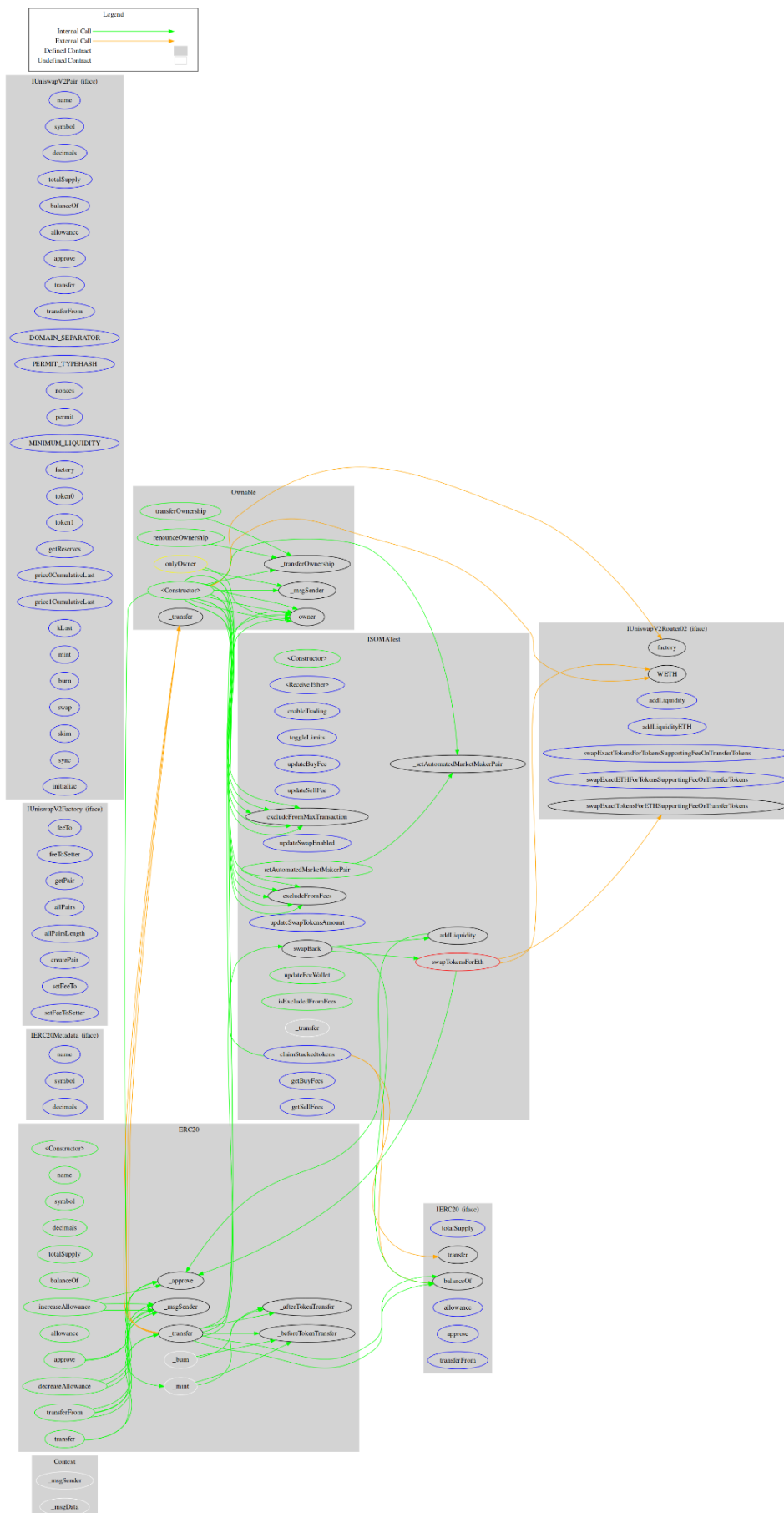
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
<b>ISOMATest</b>	Implementation	ERC20, Ownable		
		Public	✓	ERC20
		External	Payable	-
	enableTrading	External	✓	onlyOwner
	toggleLimits	External	✓	onlyOwner
	updateBuyFee	External	✓	onlyOwner
	updateSellFee	External	✓	onlyOwner
	excludeFromMaxTransaction	Public	✓	onlyOwner
	updateSwapEnabled	External	✓	onlyOwner
	claimStuckedtokens	External	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	updateSwapTokensAmount	External	✓	onlyOwner
	setAutomatedMarketMakerPair	Public	✓	onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	
	updateFeeWallet	Public	✓	onlyOwner
	isExcludedFromFees	Public		-

	_transfer	Internal	✓	
	swapTokensForEth	Private	✓	
	swapBack	Private	✓	
	addLiquidity	Private	✓	
	getBuyFees	External		-
	getSellFees	External		-

## Inheritance Graph



# Flow Graph



## Summary

ISOMA 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. Temporarily locking the contract will eliminate all the contract threats. There is also a limit of max 10% 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>