Cyberscope

Audit Report

Motion AI

April 2024

Network	ETH
Address	0x2F3208A7A8cB958f5593312ED32Be68534EE5d70
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	Unresolved
●	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
●	PMRM	Potential Mocked Router Manipulation	Unresolved
●	PVC	Price Volatility Concern	Unresolved
●	RSW	Redundant Storage Writes	Unresolved
●	RSD	Redundant Swap Duplication	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L18	Multiple Pragma Directives	Unresolved
●	L19	Stable Compiler Version	Unresolved

Table of Contents

Analysis	**1**
Diagnostics	**2**
Table of Contents	**3**
Review	**5**
Audit Updates	5
Source Files	5
Findings Breakdown	**6**
ST - Stops Transactions	7
Description	7
Recommendation	7
ELFM - Exceeds Fees Limit	8
Description	8
Recommendation	8
ZD - Zero Division	10
Description	10
Recommendation	10
PMRM - Potential Mocked Router Manipulation	11
Description	11
Recommendation	11
PVC - Price Volatility Concern	13
Description	13
Recommendation	13
RSW - Redundant Storage Writes	15
Description	15
Recommendation	15
RSD - Redundant Swap Duplication	17
Description	17
Recommendation	17
L04 - Conformance to Solidity Naming Conventions	18
Description	18
Recommendation	19
L09 - Dead Code Elimination	20
Description	20
Recommendation	20
L14 - Uninitialized Variables in Local Scope	21
Description	21
Recommendation	21
L16 - Validate Variable Setters	22
Description	22

Recommendation 22

L18 - Multiple Pragma Directives 23

Description 23

Recommendation 23

L19 - Stable Compiler Version 24

Description 24

Recommendation 24

Functions Analysis 25

Inheritance Graph 32

Flow Graph 33

Summary 34

Disclaimer 35

About Cyberscope 36

Review

Contract Name	MotionAi
Compiler Version	v0.8.25+commit.b61c2a91
Optimization	200 runs
Explorer	https://etherscan.io/address/0x2f3208a7a8cb958f5593312ed32be68534ee5d70
Address	0x2f3208a7a8cb958f5593312ed32be68534ee5d70
Network	ETH
Symbol	MotionAi
Decimals	18
Total Supply	10,000,000
Badge Eligibility	Yes

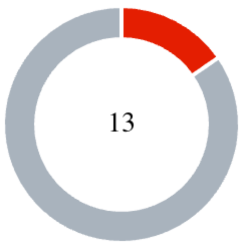
Audit Updates

Initial Audit	06 Apr 2024

Source Files

Filename	SHA256
MotionAi.sol	ba2ee1808451d8174da82714640bb68e22d832151d236dbece756b03dfe8a89b

Findings Breakdown



🟥	Critical	2
🟧	Medium	0
●	Minor / Informative	11

Severity		Unresolved	Acknowledged	Resolved	Other
🟥	Critical	2	0	0	0
🟧	Medium	0	0	0	0
●	Minor / Informative	11	0	0	0

ST - Stops Transactions

Criticality	Critical
Status	Unresolved

Description

Additionally, the contract owner has the authority to stop transactions, as described in detail in the sections `PMRM` and `ZD`. As a result, the contract might operate as a honeypot.

Recommendation

It is recommended to implement the corresponding recommendation of the `PMRM` and `ZD` findings to migrate the stop transaction finding.

ELFM - Exceeds Fees Limit

Criticality	Minor / Informative
Location	MotionAi.sol#L1284,1294
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 `setBuyTaxFeePercent` or `setSellTaxFeePercent` function with a high percentage value.

```
   function setBuyTaxFeePercent(uint256 fee) external onlyOwner
{
        require(fee <= 30, "Total Fee exceeds 30%");
        emit UpdateBuyTax(buyTax, fee);
        buyTax = fee;
    }

    function setSellTaxFeePercent(uint256 fee) external
onlyOwner {
        require(fee <= 30, "Total Fee exceeds 30%");
        emit UpdateSellTax(sellTax, fee);
        sellTax = fee;
    }
```

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.

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.

ZD - Zero Division

Criticality	Critical
Location	MotionAi.sol#L1237
Status	Unresolved

Description

The contract uses variables that can lead to transaction revert when they are combined. If the sum of `liquidityTax + marketingTax` is zero it will revert as denominator, if the sum of `liquidityTax + marketingTax` is more than `2**256` then it will revert as overflow. This can lead to unpredictable and potentially harmful results.

```
uint256 swapTokens = (contractTokenBalance * liquidityTax) / (liquidityTax +
marketingTax);
```

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 and non-overflow 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.

PMRM - Potential Mocked Router Manipulation

Criticality	Minor / Informative
Location	MotionAi.sol#L1223
Status	Unresolved

Description

The contract includes a method that allows the owner to modify the router address and create a new pair. While this feature provides flexibility, it introduces a security threat. The owner could set the router address to any contract that implements the router's interface, potentially containing malicious code. In the event of a transaction triggering the swap functionality with such a malicious contract as the router, the transaction may be manipulated.

```
    function updateUniswapV2Router(address newAddress) public
onlyOwner {
        require(
            newAddress != address(uniswapV2Router),
            "The router already has that address"
        );
        emit UpdateUniswapV2Router(newAddress,
address(uniswapV2Router));
        uniswapV2Router = IUniswapV2Router02(newAddress);
    }
```

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.

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.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	MotionAi.sol#L1380
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.

```
uint256 contractTokenBalance = balanceOf(address(this));
bool canSwap = contractTokenBalance >= swapTokensAtAmount
if (
    swapAndLiquifyEnabled &&
    canSwap &&
    !swapping &&
    !automatedMarketMakerPairs[from] &&
    from != liquidityWallet &&
    to != liquidityWallet
) {
    swapping = true
    uint256 swapTokens = (contractTokenBalance * liquidityTax)
/
        (liquidityTax + marketingTax);
    swapAndLiquify(swapTokens)
    uint256 sellTokens = balanceOf(address(this));
    swapAndSendToMarketing(sellTokens);
```

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 exchange reserves. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	MotionAi.sol#L1253,1262,1272
Status	Unresolved

Description

The contract modifies the state of the following variables without checking if their current value is the same as the one given as an argument. As a result, the contract performs redundant storage writes, when the provided parameter matches the current state of the variables, leading to unnecessary gas consumption and inefficiencies in contract execution.

```
   function changeMarketingWallet(address _marketing) external
onlyOwner {
        emit UpdateMarketingWallet(marketingWallet,
_marketing);
        marketingWallet = _marketing;
    }

    function changeLiquidityWallet(address _liquidity) external
onlyOwner {
        emit UpdateLiquidityWallet(liquidityWallet,
_liquidity);
        liquidityWallet = _liquidity;
    }

    function excludeFromTax(
        address _user,
        bool _isExcludeFromTax
    ) external onlyOwner {
        isExcludeFromTax[_user] = _isExcludeFromTax;
        emit ExcludeFromTax(_user, _isExcludeFromTax);
    }
```

Recommendation

The team is advised to implement additional checks within to prevent redundant storage writes when the provided argument matches the current state of the variables. By incorporating statements to compare the new values with the existing values before proceeding with any state modification, the contract can avoid unnecessary storage operations, thereby optimizing gas usage.

RSD - Redundant Swap Duplication

Criticality	Minor / Informative
Location	MotionAi.sol#L1394,1396
Status	Unresolved

Description

The contract contains multiple swap methods that individually perform token swaps and transfer promotional amounts to specific addresses and features. This redundant duplication of code introduces unnecessary complexity and increases dramatically the gas consumption. By consolidating these operations into a single swap method, the contract can achieve better code readability, reduce gas costs, and improve overall efficiency.

```
swapAndLiquify(swapTokens);
...
swapAndSendToMarketing(sellTokens)
```

Recommendation

A more optimized approach could be adopted to perform the token swap operation once for the total amount of tokens and distribute the proportional amounts to the corresponding addresses, eliminating the need for separate swaps.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	MotionAi.sol#L812,813,830,850,1102,1113,1125,1126,1203
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 (uint);
    function WETH() external pure returns (address);
    address _marketing
    address _liquidity
    address _user,
    bool _isExcludeFromTax
    function setSwapAndLiquifyEnabled(bool _enabled) public
onlyOwner {
...
```

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	MotionAi.sol#L196
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 _contextSuffixLength() internal view virtual returns
(uint256) {
    return 0;
}
```

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.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	MotionAi.sol#L1254
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
uint256 fees;
```

Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	MotionAi.sol#L1105,1116
Status	Unresolved

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
marketingWallet = _marketing;
liquidityWallet = _liquidity;
```

Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

L18 - Multiple Pragma Directives

Criticality	Minor / Informative
Location	MotionAi.sol#L10,175,206,308,390,418,736,776
Status	Unresolved

Description

If the contract includes multiple conflicting pragma directives, it may produce unexpected errors. To avoid this, it's important to include the correct pragma directive at the top of the contract and to ensure that it is the only pragma directive included in the contract.

```
pragma solidity ^0.8.20;
```

Recommendation

It is important to include only one pragma directive at the top of the contract and to ensure that it accurately reflects the version of Solidity that the contract is written in.

By including all required compiler options and flags in a single pragma directive, the potential conflicts could be avoided and ensure that the contract can be compiled correctly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	MotionAi.sol#L10,175,206,308,390,418,736,776
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 ^0.8.20;
...
```

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**
IERC20Errors	Interface			
IERC721Errors	Interface			
IERC1155Errors	Interface			
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
	_contextSuffixLength	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	

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, IERC20Errors		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-

	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	_transfer	Internal	✓	
	_update	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
ERC20Burnable	Implementation	Context, ERC20		
	burn	Public	✓	-
	burnFrom	Public	✓	-
IUniswapV2Factory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-

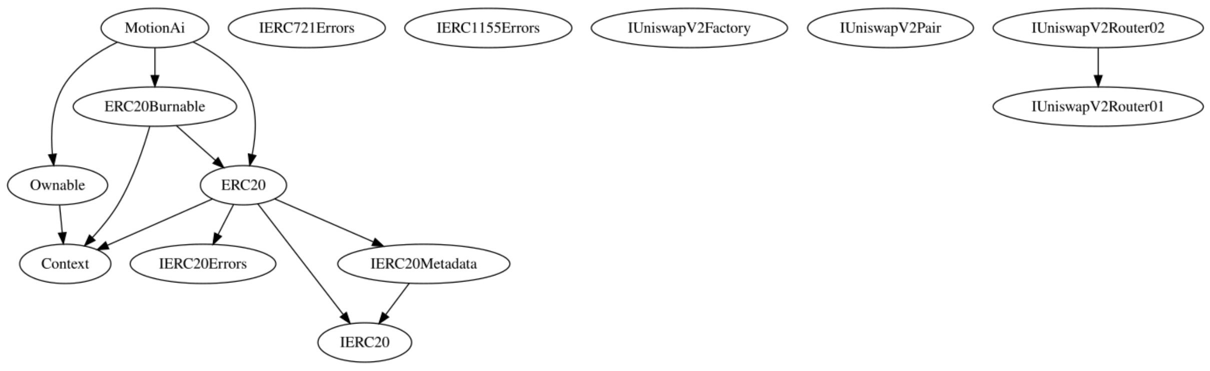
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
IUniswapV2Pair	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	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	✓	-
IUniswapV2Router01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-

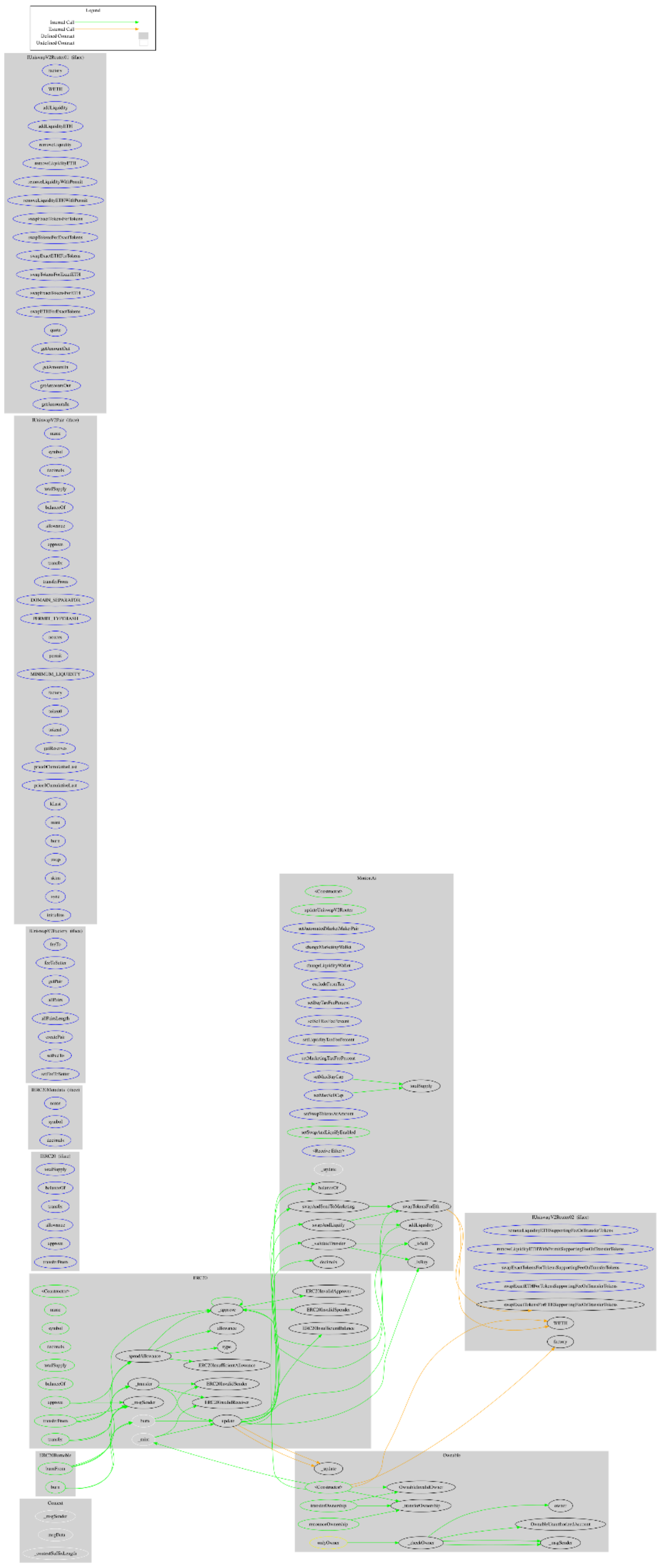
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
IUniswapV2Router02	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
MotionAi	Implementation	ERC20, ERC20Burnable, Ownable		
		Public	✓	ERC20 Ownable
	updateUniswapV2Router	Public	✓	onlyOwner
	setAutomatedMarketMakerPair	External	✓	onlyOwner
	changeMarketingWallet	External	✓	onlyOwner
	changeLiquidityWallet	External	✓	onlyOwner
	excludeFromTax	External	✓	onlyOwner

	setBuyTaxFeePercent	External	✓	onlyOwner
	setSellTaxFeePercent	External	✓	onlyOwner
	setLiquidityTaxFeePercent	External	✓	onlyOwner
	setMarketingTaxFeePercent	External	✓	onlyOwner
	setMaxBuyCap	External	✓	onlyOwner
	setMaxSellCap	External	✓	onlyOwner
	setSwapTokensAtAmount	External	✓	onlyOwner
	setSwapAndLiquifyEnabled	Public	✓	onlyOwner
		External	Payable	-
	_update	Internal	✓	
	_isSell	Internal		
	_isBuy	Internal		
	swapAndLiquify	Private	✓	
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	swapAndSendToMarketing	Private	✓	
	_validateTransfer	Private		

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IUniswapV2Router01 (iface)
IUniswapV2Pair (iface)
IUniswapV2Factory (iface)
IERC20Metadata (iface)
IERC20 (iface)
ERC20
ERC20Burnable
Context
MotionAi
IUniswapV2Router02 (iface)
Ownable

Summary

Motion 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 and 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

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