Cyberscope

Audit Report

Airtok

August 2024

Network BSC

Address 0x461ab4D656BC22230A650d9d37E1c85F90BFeD02

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
	CR	Code Repetition	Unresolved
	DDP	Decimal Division Precision	Unresolved
	MVN	Misleading Variables Naming	Unresolved
	PAMAR	Pair Address Max Amount Restriction	Unresolved
	PLPI	Potential Liquidity Provision Inadequacy	Unresolved
	PNR	Privileges Not Revoked	Unresolved
	RCS	Redundant Code Segments	Unresolved
	RRA	Redundant Repeated Approvals	Unresolved
	RSML	Redundant SafeMath Library	Unresolved
	RSD	Redundant Swap Duplication	Unresolved
	UDO	Unnecessary Decimals Override	Unresolved
	UZA	Unnecessary Zero Addition	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L13	Divide before Multiply Operation	Unresolved

Table of Contents

Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation**: This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation**: This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical**: Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium**: Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor**: Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative**: Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

Severity	Likelihood / Impact of Exploitation
Critical	Highly Likely / High Impact
Medium	Less Likely / High Impact or Highly Likely/ Lower Impact
Minor / Informative	Unlikely / Low to no Impact

Review

Contract Name	Airtok
Compiler Version	v0.8.25+commit.b61c2a91
Optimization	200 runs
Explorer	https://bscscan.com/address/0x461ab4d656bc22230a650d9d37e1c85f90bfed02
Address	0x461ab4d656bc22230a650d9d37e1c85f90bfed02
Network	BSC
Symbol	AIRTOK
Decimals	18
Total Supply	1,000,000,000
Badge Eligibility	Yes

Audit Updates

Initial Audit	13 Aug 2024

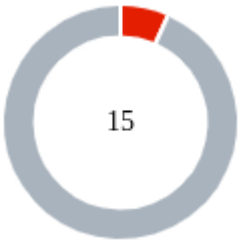
Source Files

Filename	SHA256
draft-IERC6093.sol	4aea87243e6de38804bf8737bf86f750443d3b5e63dd0fd0b7ad92f77cdbd3e3
Token.sol	00f54186c83c5d7654ae11329d313eaf82c90577d9d434d3e267e40145b37294

SafeERC20Remastered.sol	f12be6e1cb3808cd7c48764f6684435b97a62a4e5ab500c528bd2e8fc9af0be3
Ownable2Step.sol	90f1f1cdd07ce4b90e987065e82899fdaa6ef967d1996915143c6e39818e160c
Ownable.sol	5557ddc8af9d76ecc291d9fe0ad6175e09483a990726ee41697ca2a854fb3af4
Initializable.sol	d8bdbc02610707a06c0f9ab4574061df7acd8efcd398ca7d372808c4f4a0f391
IUniswapV2Router02.sol	a2900701961cb0b6152fc073856b972564f7c798797a4a044e83d2ab8f0e8d38
IUniswapV2Router01.sol	0439ffe0fd4a5e1f4e22d71ddbda76d63d61679947d158cba4ee0a1da60cf663
IUniswapV2Pair.sol	29c75e69ce173ff8b498584700fef76bc81498c1d98120e2877a1439f0c31b5a
IUniswapV2Factory.sol	51d056199e3f5e41cb1a9f11ce581aa3e190cc982db5771ffeef8d8d1f962a0d
IERC20Metadata.sol	c54116cde6fb2353a298201a9fd85375daa7a9d34bb33433e52d72d0c833ab3d
IERC20.sol	6f2faae462e286e24e091d7718575179644dc60e79936ef0c92e2d1ab3ca3cee
ERC20Burnable.sol	dcd71cbfb559a195cd0ed395d69e4719349e08da9729555548db39fa57d1701d
ERC20.sol	b9a23053ca7916e0d304b5cc2b049803d02c9f1f79e365b8ccf8cf0498a0d37f
Context.sol	847fda5460fee70f56f4200f59b82ae622bb03c79c77e67af010e31b7e2cc5b6
CoinDividendTracker.sol	63a1babbdc111f0e375615a8262f8ddb6b0087607dc7bc3a29e912c6d284babf

Address.sol	b3710b1712637eb8c0df81912da3450da6ff67b0b3ed18146b033ed15b1aa3b9

Findings Breakdown



Critical	1
Medium	0
Minor / Informative	14

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	1	0	0	0
Medium	0	0	0	0
Minor / Informative	14	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	Token.sol#L378,L490
Status	Unresolved

Description

Trading 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.

```
function enableTrading() external onlyOwner {
        if (tradingEnabled) revert TradingAlreadyEnabled();

        tradingEnabled = true;

        emit TradingEnabled();
}
```

```
function _beforeTokenUpdate(address from, address to, uint256
amount)
        internal
        view
    {
        // Interactions with DEX are disallowed prior to
enabling trading by owner
        if (!tradingEnabled) {
            if ((AMMs[from] && !AMMs[to] &&
!isExcludedFromTradingRestriction[to]) || (AMMs[to] &&
!AMMs[from] && !isExcludedFromTradingRestriction[from])) {
                revert TradingNotEnabled();
            }
        }

    }
```

Additionally, the contract owner has the authority to stop transactions, as described in detail in section PAMAR. As a result, the contract might operate as a honeypot.

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.

CR - Code Repetition

Criticality	Minor / Informative
Location	Token.sol#L225,L236,L281,311
Status	Unresolved

Description

The contract contains repetitive code segments. There are potential issues that can arise when using code segments in Solidity. Some of them can lead to issues like gas efficiency, complexity, readability, security, and maintainability of the source code. It is generally a good idea to try to minimize code repetition where possible.

```
function operationsFeesSetup(uint16 _buyFee, uint16 _sellFee,
uint16 _transferFee) public onlyOwner {
        totalFees[0] = totalFees[0] - operationsFees[0] +
_buyFee;
        totalFees[1] = totalFees[1] - operationsFees[1] +
_sellFee;
        totalFees[2] = totalFees[2] - operationsFees[2] +
_transferFee;
        ...
}
```

```
function autoBurnFeesSetup(uint16 _buyFee, uint16 _sellFee,
uint16 _transferFee) public onlyOwner {
        totalFees[0] = totalFees[0] - autoBurnFees[0] +
_buyFee;
        totalFees[1] = totalFees[1] - autoBurnFees[1] +
_sellFee;
        totalFees[2] = totalFees[2] - autoBurnFees[2] +
_transferFee;
        ...
}
```

```
function liquidityFeesSetup(uint16 _buyFee, uint16 _sellFee,
uint16 _transferFee) public onlyOwner {
        totalFees[0] = totalFees[0] - liquidityFees[0] +
_buyFee;
        totalFees[1] = totalFees[1] - liquidityFees[1] +
_sellFee;
        totalFees[2] = totalFees[2] - liquidityFees[2] +
_transferFee;
        ...
}
```

```
function rewardsFeesSetup(uint16 _buyFee, uint16 _sellFee,
uint16 _transferFee) public onlyOwner {
        totalFees[0] = totalFees[0] - rewardsFees[0] + _buyFee;
        totalFees[1] = totalFees[1] - rewardsFees[1] +
_sellFee;
        totalFees[2] = totalFees[2] - rewardsFees[2] +
_transferFee;
        ...
}
```

Recommendation

The team is advised to avoid repeating the same code in multiple places, which can make the contract easier to read and maintain. The authors could try to reuse code wherever possible, as this can help reduce the complexity and size of the contract. For instance, the contract could reuse the common code segments in an internal function in order to avoid repeating the same code in multiple places.

DDP - Decimal Division Precision

Criticality	Minor / Informative
Location	Token.sol#L421
Status	Unresolved

Description

Division of decimal (fixed point) numbers can result in rounding errors due to the way that division is implemented in Solidity. Thus, it may produce issues with precise calculations with decimal numbers.

Solidity represents decimal numbers as integers, with the decimal point implied by the number of decimal places specified in the type (e.g. decimal with 18 decimal places). When a division is performed with decimal numbers, the result is also represented as an integer, with the decimal point implied by the number of decimal places in the type. This can lead to rounding errors, as the result may not be able to be accurately represented as an integer with the specified number of decimal places.

Hence, the splitted shares will not have the exact precision and some funds may not be calculated as expected.

```
_operationsPending += fees * operationsFees[txType] /
totalFees[txType];

if (autoBurnFees[txType] > 0) {
   autoBurnPortion = fees * autoBurnFees[txType] /
totalFees[txType];
   super._update(from, address(0), autoBurnPortion);
   emit AutoBurned(autoBurnPortion);
   }

   _liquidityPending += fees * liquidityFees[txType] /
totalFees[txType];

  _rewardsPending += fees * rewardsFees[txType] /
totalFees[txType];
```

Recommendation

The team is advised to take into consideration the rounding results that are produced from the solidity calculations. The contract could calculate the subtraction of the divided funds in the last calculation in order to avoid the division rounding issue.

MVN - Misleading Variables Naming

Criticality	Minor / Informative
Location	Token.sol#L253
Status	Unresolved

Description

Variables can have misleading names if their names do not accurately reflect the value they contain or the purpose they serve. Misleading variable names can lead to confusion, making the code more difficult to read and understand. In this case, the variable `coinsReceived` is calculated after the execution of `_swapTokensForCoin(token2Swap)` and implies the amount of tokens received from the swap. Nevertheless, this is not a correct calculation, as it accounts for the total balance of the contract and not for the tokens swaped.

```
_swapTokensForCoin(token2Swap);
uint256 coinsReceived = address(this).balance;
```

Recommendation

It's always a good practice for the contract to contain variable names that are specific and descriptive. To correctly account for the received tokens the team should calculate the difference in the balance before and after the swap.

PAMAR - Pair Address Max Amount Restriction

Criticality	Minor / Informative
Location	Token.sol#L507
Status	Unresolved

Description

The contract is configured to enforce a maximum token accumulation limit through checks. This mechanism aims to prevent excessive token concentration by reverting transactions that overcome the specified cap. However, this functionality encounters issues when transactions default to the pair address during sales. If the pair address is not listed in the exceptions, then the sale transactions are inadvertently stopped, effectively disrupting operations and making the contract susceptible to unintended behaviors akin to a honeypot.

```
function _afterTokenUpdate(address from, address to, uint256 amount)
        internal
    {
        if (!isExcludedFromLimits[to] && balanceOf(to) > maxWalletAmount)
{
            revert CannotExceedMaxWalletAmount(maxWalletAmount);
        }

    }
```

```
function excludeFromLimits(address account, bool isExcluded) external
onlyOwner
{
        _excludeFromLimits(account, isExcluded);
}

function _excludeFromLimits(address account, bool isExcluded) internal
{
        isExcludedFromLimits[account] = isExcluded;
        emit ExcludeFromLimits(account, isExcluded);
}
```

Recommendation

It is advised to modify the contract to ensure uninterrupted operations by either permitting the pair address to exceed the established token accumulation limit or by safeguarding its status in the exception list. By recognizing and allowing these essential addresses the flexibility to hold more tokens than typical limits, the contract can maintain seamless transaction flows and uphold the liquidity and stability of the ecosystem. This modification is vital for avoiding disruptions that could impact the functionality and security of the contract.

PLPI - Potential Liquidity Provision Inadequacy

Criticality	Minor / Informative
Location	Token.sol#L188
Status	Unresolved

Description

The contract operates under the assumption that liquidity is consistently provided to the pair between the contract's token and the native currency. However, there is a possibility that liquidity is provided to a different pair. This inadequacy in liquidity provision in the main pair could expose the contract to risks. Specifically, during eligible transactions, where the contract attempts to swap tokens with the main pair, a failure may occur if liquidity has been added to a pair other than the primary one. Consequently, transactions triggering the swap functionality will result in a revert.

```
function _swapTokensForCoin(uint256 tokenAmount) private {
        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = routerV2.WETH();

        _approve(address(this), address(routerV2), tokenAmount);


routerV2.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount
, 0, path, address(this), block.timestamp);
}
```

Recommendation

The team is advised to implement a runtime mechanism to check if the pair has adequate liquidity provisions. This feature allows the contract to omit token swaps if the pair does not have adequate liquidity provisions, significantly minimizing the risk of potential failures.

Furthermore, the team could ensure the contract has the capability to switch its active pair in case liquidity is added to another pair.

Additionally, the contract could be designed to tolerate potential reverts from the swap functionality, especially when it is a part of the main transfer flow. This can be achieved by executing the contract's token swaps in a non-reversible manner, thereby ensuring a more resilient and predictable operation.

PNR - Privileges Not Revoked

Criticality	Minor / Informative
Location	Token.sol#L195
Status	Unresolved

Description

The contract grants specific addresses certain privileges, providing operational flexibility. However, the current implementation does not automatically remove these privileges when an address's status changes. Consequently, the privileges remain active for addresses that no longer hold the specified roles or ownership. This oversight can lead to unintended discrepancies in privileges and poses potential security risks.

```
function operationsAddressSetup(address _newAddress) public onlyOwner {
        if (_newAddress == address(0)) revert
InvalidTaxRecipientAddress(address(0));

        operationsAddress = _newAddress;
        excludeFromFees(_newAddress, true);
        _excludeFromLimits(_newAddress, true);

        emit WalletTaxAddressUpdated(1, _newAddress);
}
```

Recommendation

It is advised to modify the contract to include functionality that revokes privileges from old addresses and grants them to new ones whenever there is a change in roles or ownership. This method will ensure consistent and equitable distribution of privileges while preserving the integrity and security of the contract.

RCS - Redundant Code Segments

Criticality	Minor / Informative
Location	Token.sol#L446
Status	Unresolved

Description

The contract is implementing an if statement that contains the `false` keyword in the if condition. The inclusion of `false` does not impact the logical outcome of the condition, as the `false` value in a logical `OR (||)` operation is effectively ignored. Specifically, the `false` condition of the `OR` operation will always fail, leading the result to solely depend on the evaluation of the second condition, `_operationsPending > 0` . The presence of the `false` keyword in this context is thus redundant.

```
function _update(address from, address to, uint256 amount){
        ...
         if (false || _operationsPending > 0){
            ...
         }
        ...
}
```

Recommendation

It is recommended to remove the `false` keyword from the if statement to enhance code clarity and maintainability. Simplifying the if condition will make the code more readable and straightforward. These changes will not alter the functionality of the contract but will improve the overall quality and understandability of the code, as well as potentially reduce gas costs by eliminating unnecessary operations.

RRA - Redundant Repeated Approvals

Criticality	Minor / Informative
Location	Token.sol#L193,L267
Status	Unresolved

Description

The contract is designed to `approve` token transfers during the contract's operation by calling the _approve function before specific operations. This approach results in additional gas costs since the approval process is repeated for every operation execution, leading to inefficiencies and increased transaction expenses.

```
_approve(address(this), address(routerV2), tokenAmount);
```

Recommendation

Since the approved address is a trusted third-party source, it is recommended to optimize the contract by approving the maximum amount of tokens once in the initial set of the variable, rather than before each operation. This change will reduce the overall gas consumption and improve the efficiency of the contract.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	CoinDividendTracker.sol#L7
Status	Unresolved

Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert to underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases gas consumption unnecessarily in cases where the explanatory error message is not used.

```
library SafeMathUint {
  ...
}
```

Recommendation

The team is advised to remove the SafeMath library in cases where the revert error message is not used. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

Read more about the breaking change on https://docs.soliditylang.org/en/stable/080-breaking-changes.html#solidity-v0-8-0-breaking-changes.

RSD - Redundant Swap Duplication

Criticality	Minor / Informative
Location	Token.sol#L374
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.

```
function _update(address from, address to, uint256 amount)
        internal
        override
{
        ...

        _swapTokensForCoin(token2Swap);
        ...

        if (_liquidityPending > 0) {
            _swapAndLiquify(_liquidityPending);
            _liquidityPending = 0;
        }

        if (_rewardsPending > 0 && getNumberOfDividendTokenHolders() >
0) {
            _sendDividends(_rewardsPending);
            _rewardsPending = 0;
        }
        ...
}
```

```
function _swapAndLiquify(uint256 tokenAmount) private returns (uint256
leftover) {
        ...
        _swapTokensForCoin(halfAmount);
        ...
}
```

```
function _sendDividends(uint256 tokenAmount) private {
        _swapTokensForCoin(tokenAmount);
        ...
    }
```

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.

UDO - Unnecessary Decimals Override

Criticality	Minor / Informative
Location	Token.sol#L166
Status	Unresolved

Description

The contract is currently implementing an override of the decimals function, which simply returns the value 18. This override is redundant since the extending token contract already specifies 18 decimals as its standard. In the context of ERC-20 tokens, 18 decimals is a common default, and overriding this function to return the same value adds unnecessary complexity to the contract. This redundancy does not contribute to the functionality of the contract and could potentially lead to confusion about the necessity of this override.

```
function decimals() public pure override returns (uint8) {
        return 18;
}
```

Recommendation

Since the inherited ERC-20 contract already defines the decimals number, maintaining an overriding function that merely repeats this value does not contribute to the contract's effectiveness. As a result, it is recommended to remove the redundant `decimals` function from the contract. Removing this function will simplify the contract, making it more straightforward to maintain without impacting its operational capabilities.

UZA - Unnecessary Zero Addition

Criticality	Minor / Informative
Location	Token.sol#L211,L447
Status	Unresolved

Description

The contract is utilizing the `getAllPending()` function, designed to calculate the total amount of pending operations, liquidity and rewards fees. Within this function, the total is calculated using the expression `return 0 + _operationsPending + _liquidityPending + _rewardsPending;`. The addition of zero at the beginning of this expression is redundant and does not affect the outcome of the calculation. Similarly, the `_update()` function includes the statement `uint256 token2Swap = 0 + _operationsPending` which is also redundant.

In programming, adding zero to a number does not change its value, making this part of the expression superfluous. This unnecessary operation could potentially lead to confusion about the code's intent and unnecessarily clutters the expression, detracting from the overall clarity and simplicity of the contract's code.

```
function getAllPending() public view returns (uint256) {
        return 0 + _operationsPending + _liquidityPending +
_rewardsPending;
}
```

```
function _update(address from, address to, uint256 amount){
        ...
        uint256 token2Swap = 0 + _operationsPending;
        ...
}
```

Recommendation

It is recommended to remove the redundant zero addition from the expressions, to improve code clarity and conciseness. This change not only adheres to best coding practices by

eliminating unnecessary operations but also enhances the readability of the code, making it easier for other users to understand the contract's functionality. Simplifying expressions where possible is a key aspect of writing clean, efficient, and easily maintainable code in smart contract development.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Token.sol#L63,162,198,214,224,235,280,310,337,366
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.

```
mapping (address => bool) public AMMs
address _router
uint16 _swapThresholdRatio
address _newAddress
uint16 _transferFee
uint16 _sellFee
uint16 _buyFee
address AMM
uint256 _maxWalletAmount
```

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/stable/style-guide.html#naming-conventions.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	Token.sol#L418,421,424,429,431
Status	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 prediction.

```
fees = amount * totalFees[txType] / 10000
_liquidityPending += fees * liquidityFees[txType] /
totalFees[txType]
```

Recommendation

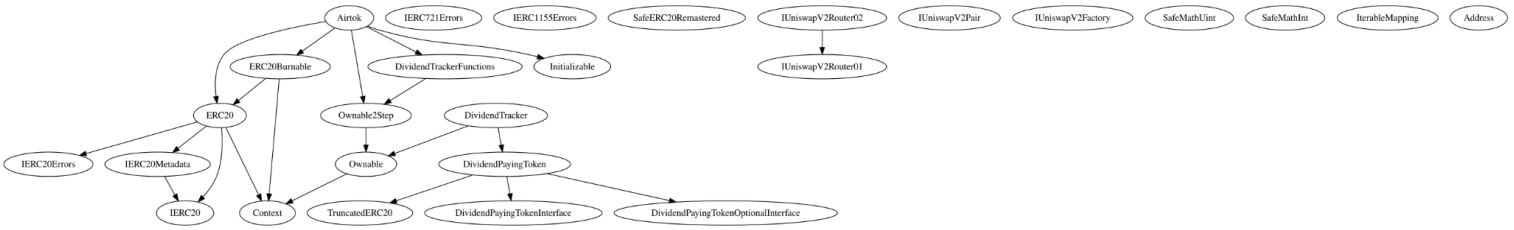
To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

Functions Analysis

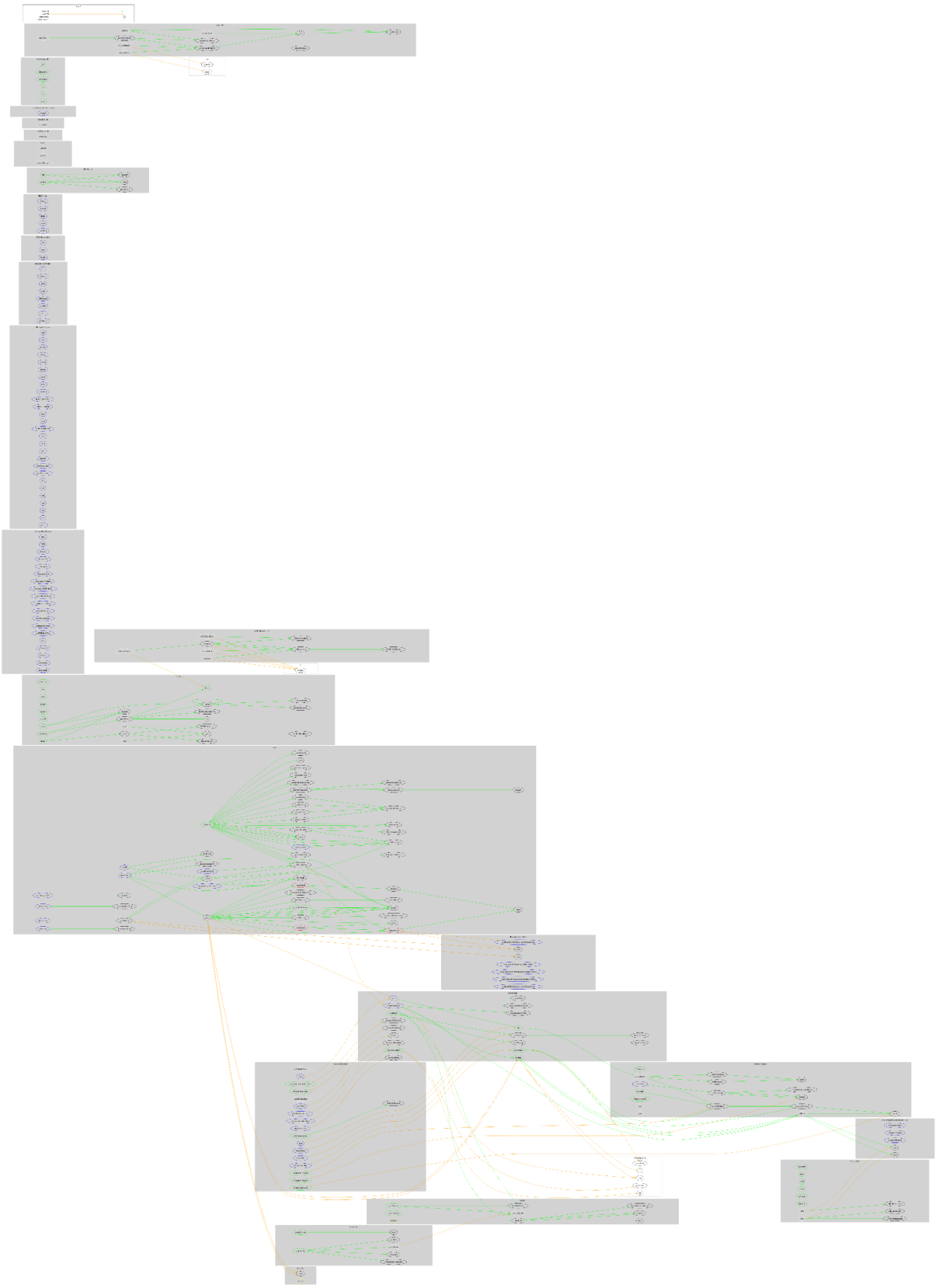
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Airtok	Implementation	ERC20, ERC20Burnable, Ownable2Step, DividendTrackerFunctions, Initializable		
		Public	✓	ERC20 Ownable
	afterConstructor	External	✓	initializer
	decimals	Public		-
	recoverToken	External	✓	onlyOwner
	recoverForeignERC20	External	✓	onlyOwner
		External	Payable	-
	_swapTokensForCoin	Private	✓	
	updateSwapThreshold	Public	✓	onlyOwner
	getSwapThresholdAmount	Public		-
	getAllPending	Public		-
	operationsAddressSetup	Public	✓	onlyOwner
	operationsFeesSetup	Public	✓	onlyOwner
	autoBurnFeesSetup	Public	✓	onlyOwner
	_swapAndLiquify	Private	✓	
	_addLiquidity	Private	✓	
	addLiquidityFromLeftoverTokens	External	✓	-

	liquidityFeesSetup	Public	✓	onlyOwner
	_sendDividends	Private	✓	
	excludeFromDividends	External	✓	onlyOwner
	_excludeFromDividends	Internal	✓	
	rewardsFeesSetup	Public	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	_updateRouterV2	Private	✓	
	setAMM	External	✓	onlyOwner
	_setAMM	Private	✓	
	excludeFromLimits	External	✓	onlyOwner
	_excludeFromLimits	Internal	✓	
	updateMaxWalletAmount	Public	✓	onlyOwner
	_maxWalletSafeLimit	Private		
	enableTrading	External	✓	onlyOwner
	excludeFromTradingRestriction	Public	✓	onlyOwner
	_update	Internal	✓	
	_beforeTokenUpdate	Internal		
	_afterTokenUpdate	Internal	✓	

Inheritance Graph



Flow Graph



Summary

Airtok contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. Airtok is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler errors, but one critical issue. The contract Owner can access some admin functions that could potentially be used in a malicious way to disturb the users' transactions. There is also a limit of max 25% 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

cyberscope.io