Audit Report

Iqra Token V2

May 2024

Network BSC

Address 0xCCe450FE71b49a72e318340693280FEA50f316d2

Audited by © cyberscope

Analysis

Critical Medium Minor / Informative Pass

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

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
Minor / Informative	CO	Code Optimization	Unresolved
Minor / Informative	DDP	Decimal Division Precision	Unresolved
Minor / Informative	PLPI	Potential Liquidity Provision Inadequacy	Unresolved
Minor / Informative	REI	Redundant ERC20 Inheritance	Unresolved
Minor / Informative	RFI	Redundant Function Implementations	Unresolved
Minor / Informative	RSW	Redundant Storage Writes	Unresolved
Minor / Informative	L04	Conformance to Solidity Naming Conventions	Unresolved
Minor / Informative	L13	Divide before Multiply Operation	Unresolved

Table of Contents

Review

Contract Name	Iqra_Token_V2
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	200 runs
Explorer	https://bscscan.com/address/0xcce450fe71b49a72e318340693280fea50f316d2
Address	0xcce450fe71b49a72e318340693280fea50f316d2
Network	BSC
Symbol	IQRA
Decimals	18
Total Supply	400,000,000,000
Badge Eligibility	Yes

Audit Updates

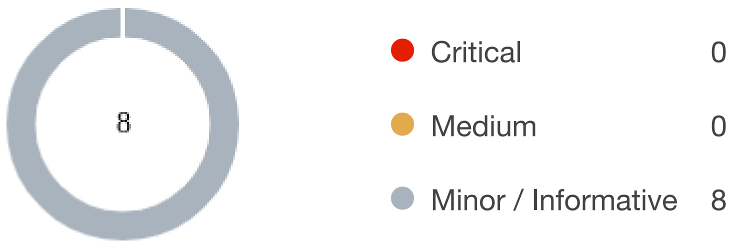
Initial Audit	24 May 2024

Source Files

Filename	SHA256
Token.sol	0483babab912808f353a15d10ae3fcbaf3680e42f84f2b5999fdb686d73bcc24
Ownable2Step.sol	3e3bdb084bc14ade54e8259e710287956a7dbf2b2b4ad1e4cd8899d2293c7241

Ownable.sol	33422e7771fefe5fbfe8934837515097119d82a50eda0e49b38e4d6a64a1c25d
Initializable.sol	b05c26d897c4178cbdb35ad113527e463e1bdeae5764869318a54f93c8b98a94
IUniswapV2Router02.sol	a2900701961cb0b6152fc073856b972564f7c798797a4a044e83d2ab8f0e8d38
IUniswapV2Router01.sol	0439ffe0fd4a5e1f4e22d71ddbda76d63d61679947d158cba4ee0a1da60cf663
IUniswapV2Pair.sol	29c75e69ce173ff8b498584700fef76bc81498c1d98120e2877a1439f0c31b5a
IUniswapV2Factory.sol	51d056199e3f5e41cb1a9f11ce581aa3e190cc982db5771ffeef8d8d1f962a0d
IERC20Metadata.sol	b10e2f8bcc3ed53a5d9a82a29b1ad3209225331bb4de4a0459862a762cf83a1a
IERC20.sol	7ebde70853ccafcf1876900dad458f46eb9444d591d39bfc58e952e2582f5587
ERC20Burnable.sol	480b22ce348050fdb85a693e38ed6b4767a94e4776fc6806d6808a0ec171177e
ERC20.sol	f70c6ae5f2dda91a37e17cfcbec390cc59515ed0d34e316f036f5431b5c0a3f2
Context.sol	b2cfee351bcafd0f8f27c72d76c054df9b571b62cfac4781ed12c86354e2a56c

Findings Breakdown



Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	8	0	0	0

CO - Code Optimization

Criticality	Minor / Informative
Location	Token.sol#L63,118,251,296
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.

- At the constructor, the `_transferOwnership` function call doesn't make use of `supplyRecipient` variable, as well as the `_mint` call has a redundant division by 10.
- At the `getAllPending` function there is a redundant zero added to the return value.
- At the `_transfer` function there is a redundant `false ||` in a boolean statement, as well as a redundant zero added to `token2Swap` variable.
- At the `_setAMMPair` function there is a redundant if statement that does nothing on true or false.

```
constructor()
    ERC20(unicode"Iqra Token V2", unicode"IQRA")
{
    address supplyRecipient =
0x92C33EEAa572cC1AcbbD00E50dba4a514bF53AC2;
    ...
    _mint(supplyRecipient, 10000000000000 * (10 ** decimals()) / 10);
    _transferOwnership(0x92C33EEAa572cC1AcbbD00E50dba4a514bF53AC2);
}
...
function getAllPending() public view returns (uint256) {
    return 0 + _mindsafePending + _liquidityPending;
}
...
function _transfer(
    address from,
    address to,
    uint256 amount
) internal override {
    ...
        if (false || _mindsafePending > 0) {
            uint256 token2Swap = 0 + _mindsafePending;
            bool success = false;

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

            uint256 mindsafePortion = coinsReceived * _mindsafePending /
token2Swap;
            if (mindsafePortion > 0) {
                success =
payable(mindsafeAddress).send(mindsafePortion);
                if (success) {
                    emit mindsafeFeeSent(mindsafeAddress,
mindsafePortion);
                }
            }
            _mindsafePending = 0;

        }
    ...

}
...
function _setAMMPair(address pair, bool isPair) private {
    AMMPairs[pair] = isPair;

    if (isPair) {
    }
```

```
    emit AMMPairsUpdated(pair, isPair);
}
```

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.

DDP - Decimal Division Precision

Criticality	Minor / Informative
Location	Token.sol#L115,225,228,231,236
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.

```
return balanceOf(pairV2) * swapThresholdRatio / 10000;
...
fees = amount * totalFees[txType] / 10000;
...
_mindsafePending += fees * mindsafeFees[txType] / totalFees[txType];
...
autoBurnPortion = fees * autoBurnFees[txType] / totalFees[txType];
...
_liquidityPending += fees * liquidityFees[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.

PLPI - Potential Liquidity Provision Inadequacy

Criticality	Minor / Informative
Location	Token.sol#L97,173,204
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);
}
...
function _addLiquidity(uint256 tokenAmount, uint256 coinAmount) private
returns (uint, uint, uint) {
    _approve(address(this), address(routerV2), tokenAmount);

    return routerV2.addLiquidityETH{value: coinAmount}(address(this),
tokenAmount, 0, 0, address(0), block.timestamp);
}
...
function _transfer(
    address from,
    address to,
    uint256 amount
) internal override {
    ...
        if (false || _mindsafePending > 0) {
            uint256 token2Swap = 0 + _mindsafePending;
            bool success = false;

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

            uint256 mindsafePortion = coinsReceived * _mindsafePending /
token2Swap;
            if (mindsafePortion > 0) {
                success =
payable(mindsafeAddress).send(mindsafePortion);
                if (success) {
                    emit mindsafeFeeSent(mindsafeAddress,
mindsafePortion);
                }
            }
            _mindsafePending = 0;

        }
    ...

}
```

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.

REI - Redundant ERC20 Inheritance

Criticality	Minor / Informative
Location	Token.sol#L22
Status	Unresolved

Description

The contract `Iqra_Token_V2` inherits from both `ERC20` and `ERC20Burnable`. The `ERC20Burnable` contract already extends the `ERC20` contract, making the direct inheritance of `ERC20` redundant. This redundancy does not introduce any functional issues or security vulnerabilities but can lead to unnecessary code duplication and increased contract size.

```
contract Iqra_Token_V2 is ERC20, ERC20Burnable, Ownable2Step,
Initializable
```

Recommendation

The team is advised to remove the direct inheritance of `ERC20` from the `Iqra_Token_V2` contract, as it is already included through the inheritance of `ERC20Burnable`.

RFI - Redundant Function Implementations

Criticality	Minor / Informative
Location	Token.sol#L305,312
Status	Unresolved

Description

The contract implements the ERC20 standard and includes the hooks `_beforeTokenTransfer` and `_afterTokenTransfer`. However, these functions only call their respective abstract functions from the parent contract, making them redundant. The implementation is as follows:

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

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

These redundant hook functions do not add any additional logic or functionality beyond calling the parent contract's functions. This can lead to unnecessary code bloat and reduce the overall readability and maintainability of the contract.

Recommendation

The team is advised to remove the `_beforeTokenTransfer` and `_afterTokenTransfer` functions if no additional logic is intended to be added. This will streamline the code and reduce confusion for future maintainers of the contract. By removing these redundant functions, the contract becomes more concise and maintainable, adhering to best practices in smart contract development.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	Token.sol#L211,297
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 excludeFromFees(address account, bool isExcluded) public
onlyOwner {
    isExcludedFromFees[account] = isExcluded;

    emit ExcludeFromFees(account, isExcluded);
}
...
function setAMMPair(address pair, bool isPair) external onlyOwner {
    require(pair != pairV2, "DefaultRouter: Cannot remove initial pair
from list");

    _setAMMPair(pair, isPair);
}
```

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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Token.sol#L22,43,47,48,49,51,52,54,55,87,107,122,131,142,187
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.

```
uint16 public swapThresholdRatio;

uint256 private _mindsafePending;
uint256 private _liquidityPending;
}


...
```

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.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	Token.sol#L225,228,231,236
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.

```
= amount * totalFees[txType] / 10000;

dsafePending += fees * mindsafeFees[txType] / totalFees[txType];
```

Recommendation

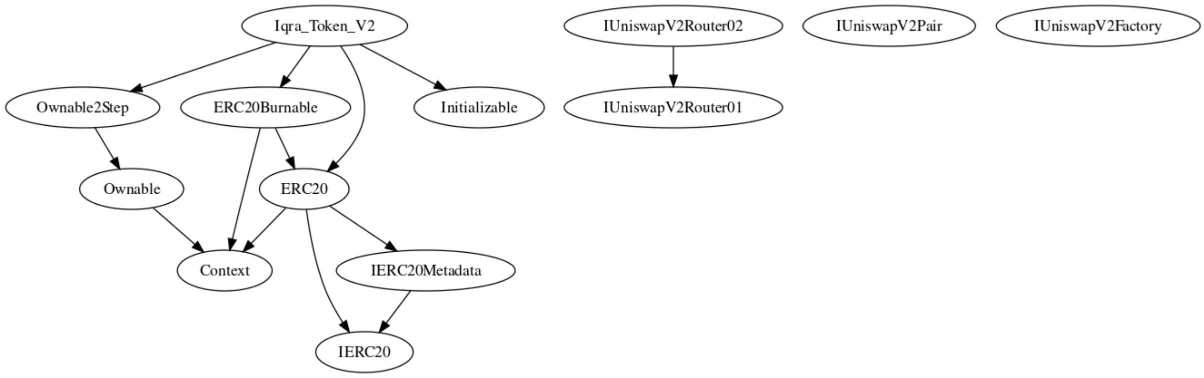
To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

Functions Analysis

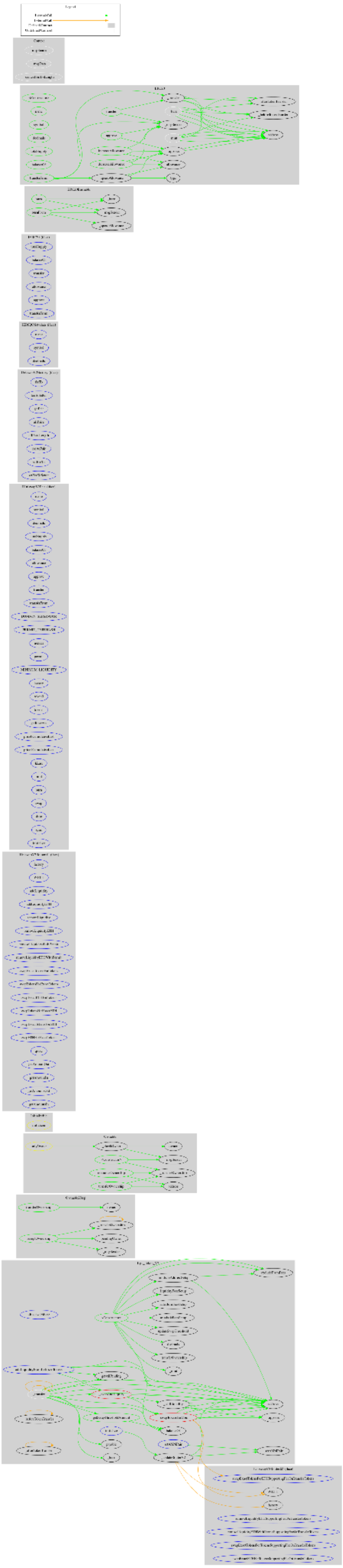
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Iqra_Token_V2	Implementation	ERC20, ERC20Burnable, Ownable2Step, Initializable		
		Public	✓	ERC20
	initialize	External	✓	initializer
		External	Payable	-
	decimals	Public		-
	_swapTokensForCoin	Private	✓	
	updateSwapThreshold	Public	✓	onlyOwner
	getSwapThresholdAmount	Public		-
	getAllPending	Public		-
	mindsafeAddressSetup	Public	✓	onlyOwner
	mindsafeFeesSetup	Public	✓	onlyOwner
	autoBurnFeesSetup	Public	✓	onlyOwner
	_swapAndLiquify	Private	✓	
	_addLiquidity	Private	✓	
	addLiquidityFromLeftoverTokens	External	✓	-
	liquidityFeesSetup	Public	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner

	_transfer	Internal	✓	
	_updateRouterV2	Private	✓	
	setAMMPair	External	✓	onlyOwner
	_setAMMPair	Private	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	

Inheritance Graph



Flow Graph



Summary

Iqra Token is an interesting project that has a friendly and growing community. There is also a limit of max 25% fees. The maximum fee percentage that can be set is 25%. A multi-wallet signing pattern will provide security against potential hacks.

This audit investigates security issues, business logic concerns and potential improvements.

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