Cyberscope

Audit Report

QCI

December 2024

Network BSC

Address 0x50B21De0d7F369F313d40150070368d338b1a926

Audited by © cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	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
●	TFB	Transfer Fee Bypass	Unresolved
●	IBM	Ineffective Burn Mechanism	Unresolved
●	MVN	Misleading Variable Naming	Unresolved
●	MEE	Missing Events Emission	Unresolved
●	RMA	Redundant Mapping Assignment	Unresolved
●	L19	Stable Compiler Version	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	QCI
Compiler Version	v0.8.27+commit.40a35a09
Optimization	200 runs
Explorer	https://bscscan.com/address/0x50b21de0d7f369f313d40150070368d338b1a926
Address	0x50b21de0d7f369f313d40150070368d338b1a926
Network	BSC
Symbol	QCI
Decimals	18
Total Supply	150,000,000
Badge Eligibility	Must Fix Criticals

Audit Updates

Initial Audit	02 Dec 2024

Source Files

Filename	SHA256
QCI.sol	ff3d44fd6c661800d88aedb44dfffef8ea34379e323524864c180a49baa3b5d7

Findings Breakdown

Critical	1
Medium	0
Minor / Informative	5

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

TFB - Transfer Fee Bypass

Criticality	Critical
Location	QCI.sol#L32,37
Status	Unresolved

Description

The contract is found to include the `_taxedTransfer` function, which is intended to apply transfer taxes by deducting a percentage of the transferred amount as fees. However, this function is only applied within the `transfer` function and not the `transferFrom` function. As a result, users executing token transfers through the `transferFrom` function can bypass the fee mechanism, leading to an inconsistent application of the transfer tax. This behaviour may result in an unintended advantage for users utilizing `transferFrom` and could undermine the intended tax structure of the contract.

```
    function transfer(address recipient, uint256 amount) public
override returns (bool) {
        _taxedTransfer(_msgSender(), recipient, amount);
        return true;
    }

    function _taxedTransfer(address sender, address recipient,
uint256 amount) internal {

    if (adminExcluded[sender] || adminExcluded[recipient] ||
sender == owner()) {
        super._transfer(sender, recipient, amount);
        return;
    }

    uint256 taxAmount = (amount * TRANSFER_TAX) / 100;

    if (taxAmount > 0) {
        super._transfer(sender, burnAddress, taxAmount);
        totalBurned += taxAmount;
    }

    uint256 amountAfterTax = amount - taxAmount;
    super._transfer(sender, recipient, amountAfterTax);
}
```

Recommendation

It is recommended to reevaluate the transfer fee implementation and consider integrating the `_taxedTransfer` function within the `transferFrom` function. This ensures that the transfer tax mechanism is applied consistently across all token transfers, maintaining the integrity of the fee model and preventing bypass scenarios.

IBM - Ineffective Burn Mechanism

Criticality	Minor / Informative
Location	QCI.sol#L8,47
Status	Unresolved

Description

The contract is designed to apply a transfer tax, where a portion of tokens (defined by `TRANSFER_TAX`) is sent to a designated `burnAddress`. While these tokens become inactive and non-transferable upon being sent to the `burnAddress`, they are not deducted from the total token supply. This approach does not represent an actual burn as per the ERC20 standard, where tokens are permanently removed from circulation by reducing the total supply. Consequently, while the tokens at the `burnAddress` are rendered inactive, they still contribute to the overall token supply, which may mislead users regarding the true circulating supply.

```
   address public constant burnAddress =
0x000000000000000000000000000000000000dEaD;
    uint256 public constant TRANSFER_TAX = 5;

    ...
    function _taxedTransfer(address sender, address recipient,
uint256 amount) internal {
    ...
    uint256 taxAmount = (amount * TRANSFER_TAX) / 100;

    if (taxAmount > 0) {
        super._transfer(sender, burnAddress, taxAmount);
        totalBurned += taxAmount;
    }
```

Recommendation

It is recommended to consider calling the `_burn` function of the ERC20 token standard to ensure that tokens intended for burning are permanently removed from the total supply. This will align the functionality with user expectations and standard practices, providing clarity on the actual circulating supply of the token.

MVN - Misleading Variable Naming

Criticality	Minor / Informative
Location	QCI.sol#L37
Status	Unresolved

Description

Variables can have misleading names if their names do not accurately reflect the value they contain or the purpose they serve. The contract uses some variable names that are too generic or do not clearly convey the information stored in the variable. Misleading variable names can lead to confusion, making the code more difficult to read and understand.

The contract uses the variable name `adminExcluded` to represent addresses that are exempt from transfer fees. However, the name is misleading as it implies a relationship to administrative privileges or functionality, which is not the case. This can cause confusion for developers, auditors, and users reviewing the code, as the variable's actual purpose is to exclude specific addresses from fee deductions rather than being related to administrative roles or permissions.

```
    function _taxedTransfer(address sender, address recipient,
uint256 amount) internal {

        if (adminExcluded[sender] || adminExcluded[recipient] ||
sender == owner()) {
            super._transfer(sender, recipient, amount);
            return;
        }

        uint256 taxAmount = (amount * TRANSFER_TAX) / 100;

        if (taxAmount > 0) {
            super._transfer(sender, burnAddress, taxAmount);
            totalBurned += taxAmount;
        }

        uint256 amountAfterTax = amount - taxAmount;
        super._transfer(sender, recipient, amountAfterTax);
    }
```

Recommendation

It's always a good practice for the contract to contain variable names that are specific and descriptive. The team is advised to keep in mind the readability of the code.

It is recommended to rename the `adminExcluded` variable to a more descriptive and accurate name, to clearly convey its intended functionality. This change will improve code readability and reduce potential misunderstandings regarding its purpose.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	QCI.sol#L24
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.

```
    function setAdminExcluded(address account, bool excluded)
external onlyOwner {
        adminExcluded[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.

RMA - Redundant Mapping Assignment

Criticality	Minor / Informative
Location	QCI.sol#L17
Status	Unresolved

Description

In the constructor, the `adminExcluded` mapping is assigned `true` for `msg.sender`, which is the owner of the contract. However, the `_taxedTransfer` function already includes a conditional check (`sender == owner`) that exempts the owner from transfer taxes. This makes the explicit assignment in the `adminExcluded` mapping redundant, as it does not provide additional functionality.

```
   constructor(uint256 initialSupply)
ERC20("QuantumCircuits.Inc", "QCI") Ownable(msg.sender) {
        _mint(msg.sender, initialSupply);
        adminExcluded[msg.sender] = true;
    }

    function _taxedTransfer(address sender, address recipient,
uint256 amount) internal {

    if (adminExcluded[sender] || adminExcluded[recipient] ||
sender == owner()) {
        super._transfer(sender, recipient, amount);
        return;
    }
    ...
    }
```

Recommendation

It is recommended to remove the redundant `adminExcluded[msg.sender] = true` assignment in the constructor to streamline the code and avoid unnecessary redundancy. This will simplify the logic without impacting the functionality, as the owner is already exempted from fees through the existing conditional check in `_taxedTransfer`.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	QCI.sol#L2
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 ^0.8.0;
```

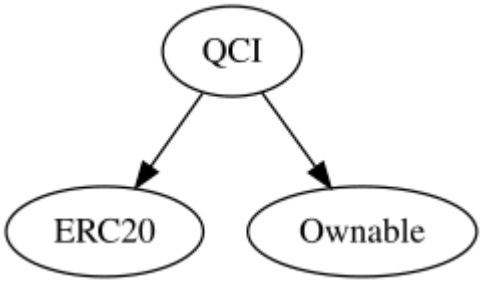
Recommendation

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.

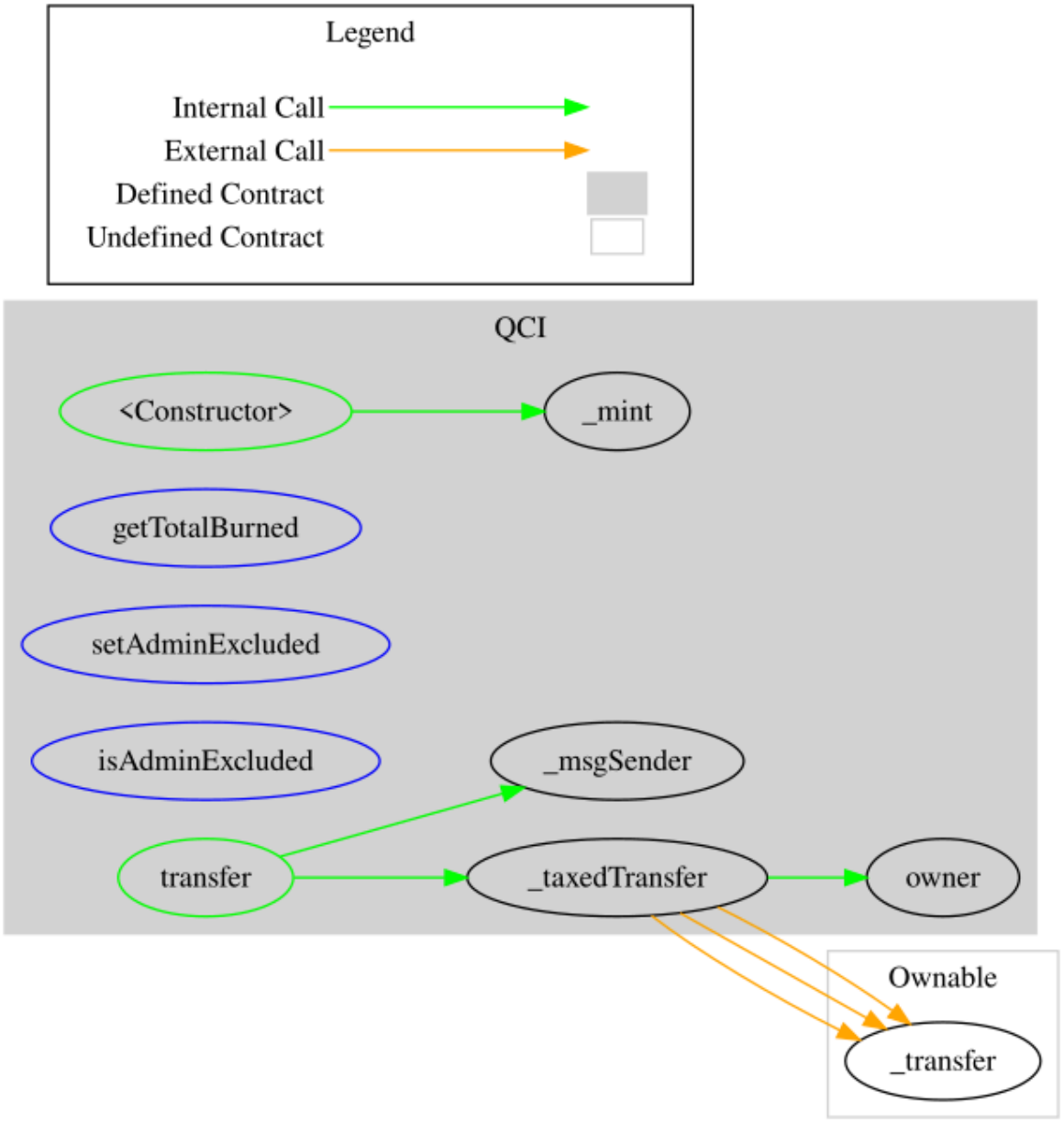
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
QCI	Implementation	ERC20, Ownable		
		Public	✓	ERC20 Ownable
	getTotalBurned	External		-
	setAdminExcluded	External	✓	onlyOwner
	isAdminExcluded	External		-
	transfer	Public	✓	-
	_taxedTransfer	Internal	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
QCI
<Constructor>
_mint
getTotalBurned
setAdminExcluded
isAdminExcluded
_msgSender
transfer
_taxedTransfer
owner
Ownable
_transfer

Summary

QCI contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. QCI is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error but 1 critical issue. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. There is also a fee of 5%.

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