QuillAudits

Audit Report
April, 2022

For

CYBER CITY

Table of Content

Executive Summary 01

Checked Vulnerabilities 03

Techniques and Methods 04

Manual Testing 05

A. Contract - RBAC, ExceptionalList, CyberCityToken 05

High Severity Issues 05

Medium Severity Issues 05

Low Severity Issues 05

A.1 | Possible to mint more than the set maximum supply of the tokens 05

Informational Issues 05

A.2 | Unlocked Pragma 06

A.3 | Unindexed event parameters 06

A.4 | Public functions that could be declared external 07

Functional Testing 08

Automated Testing 08

Closing Summary 09

About QuillAudits 10

Executive Summary

Project Name	CYBR Token
Overview	The provided code is a token contract with the following functionality and specifications: - minting of tokens is allowed only to certain addresses with special privileges. - payment of commission when transferring tokens between wallets (2.5% is burned, 2.5% goes to the project wallet) - Commissions can be reset when adding a wallet to the white list **Token Network:** Fantom blockchain **Decimals:** 18 **Total Supply:** 300 000 000
Timeline	April 20, 2022 - April 25, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyse CYBR Token codebase for quality, security, and correctness.
Github	*https://github.com/Alium-Finance/CyberCity-contracts/...CyberCityToken.sol*
Commit Hash	6e30966
Fixed In	ba2e7a868ef7434815cd2f317b9c967af5bb28fb

High
Medium
Low
Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	1	3

Types of Severities

High

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility leve

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

A. Contract - RBAC, ExceptionalList, CyberCityToken

High Severity Issues

No issues were found

Medium Severity Issues

No issues were found

Low Severity Issues

A.1 Possible to mint more than the set maximum supply of the tokens

Line 21

```
constructor(address _premintRecipient, uint256 _premintAmount, address _admin, address _dev)
    ERC20("CyberCity Token", "CYBR")
    RBAC(_admin)
{
    devFee = 250; // 2,5%
    burnFee = 250; // 2,5%
    devFeeTo = _dev;
    if (!isMember(_dev)) {
        _members[_dev] = true;
    }
    if (!isMember(_premintRecipient)) {
        _members[_premintRecipient] = true;
    }

    _mint(_premintRecipient, _premintAmount);
}
```

Description

The contract is designed in such a way that it prevents the owner from minting more than MAX_SUPPLY (i.e 300000000). If the owner tries to mint more tokens than that, the contract won't allow and the transaction would fail. However, in the constructor, there is functionality to mint a certain amount of tokens to a specific _premintRecipient address. Here, there is no check regarding how much amount can be minted to _premintRecipient. Hence if the _premintAmount is passed as 100M, 100M tokens will be minted.

Remediation

It is recommended to review the logic of the contract and implement appropriate checks to avoid this kind of issue.

Status

Fixed

Informational Issues

A.2 Unlocked pragma

Description

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Remediation

Here all the in-scope contracts have an unlocked pragma, it is recommended to lock the same.

Status

Fixed

A.3 Unindexed event parameters

Line 8, 9	`event MemberAdded(address member);` `event MemberRemoved(address member);`

17,18,19	`event BurnFeeChanged(uint256 percent);`
	`event DevFeeChanged(uint256 percent);`
	`event DevFeeToChanged(address feeTo);`

Description

MemberAdded, MemberRemoved, BurnFeeChanged, DevFeeChanged, DevFeeToChanged events don't have any indexed parameters. Unindexed

Remediation

Consider indexing event parameters to avoid the task of off-chain services searching and filtering for specific events.

Status

Fixed

A.4 Public functions that could be declared external inorder to save gas

Description

Whenever a function is not called internally, it is recommended to define them as external instead of public in order to save gas. For all the public functions, the input parameters are copied to memory automatically, and it costs gas. If your function is only called externally, then you should explicitly mark it as external. External function's parameters are not copied into memory but are read from calldata directly. This small optimization in your solidity code can save you a lot of gas when the function input parameters are huge.

Here is a list of functions that could be declared external:

-mint()
-burn()
-getRoleHash()

Status

Fixed

Functional Testing

Some of the tests performed are mentioned below

- should test all getters
- Should premint provided amount
- Should test getRoleHash
- Should test Access controls on adding and removing members
- Should test add and remove members functionality
- Should prevent minter to mint more than total supply
- Should test mint and burn
- should validate estimated output
- Should test setBurnFee, setDevFee, setDevFeeTo
- Should test transfer and transferFrom
- Should test fee calculation mechanism

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of the CYBR Token. We performed our audit according to the procedure described above.

Some issues of Low and informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture.

At the end, CYBR Team resolved all issues

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the CYBR Token Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the CYBR Token Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

500+
Audits Completed

$15B
Secured

500K
Lines of Code Audited

Follow Our Journey

Audit Report
April, 2022

For

CYBER CITY

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com