QuillAudits

Audit Report
February, 2024

For

ccfound

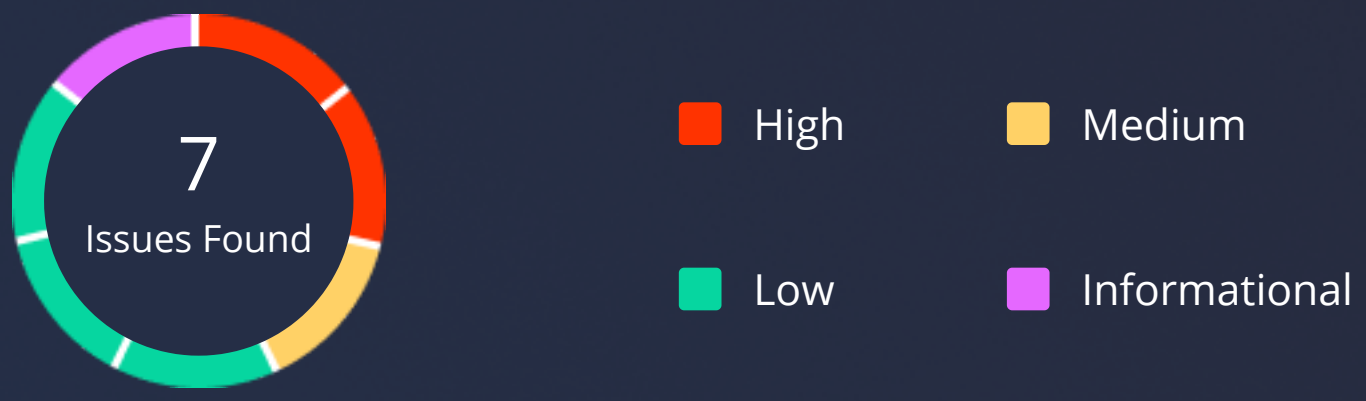
Table of Content

Table of Content

Executive Summary

Project Name	CCFound
Project URL	*https://ccfound.com/en*
Overview	CCFound Token ERC20 Contract,ccFound token is a cryptocurrency token used on the ccFOUND platform, which aims to create a decentralized knowledge and wisdom market on the internet. The platform is designed to be censorship-proof and offers various monetization methods from information products to questions with rewards.
Audit Scope	*https://bscscan.com/address/0x1acbc7d9c40aa8955281ae29b581bcf002eeb4f4#code*
Language	Solidity
Blockchain	BSC
Method	Manual Review, Automated Tools, Functional Testing
Review 1	14th February 2024 - 20th February 2024
Updated Code Received	NA
Review 2	21st February 2024
Fixed In	NA

Number of Security Issues per Severity



	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	2	1	3	1

Checked Vulnerabilities

- Access Management
- Arbitrary write to storage
- Centralization of control
- Ether theft
- Improper or missing events
- Logical issues and flaws
- Arithmetic Correctness
- Race conditions/front running
- SWC Registry
- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Malicious libraries
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- ERC's conformance
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls
- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Multiple Sends
- Using suicide
- Using delegatecall
- Upgradeable safety
- Using throw

Checked Vulnerabilities

- Using inline assembly
- Style guide violation
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

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

- The overall quality of code.
- Use of best practices.
- Code documentation and comments, match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.
- Implementation of ERC 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 analyzed 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

A 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 analyzed, 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 behavior 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, Solhint, Mythril, Slither, Solidity Static Analysis.

Types of Severity

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

High Severity Issues

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 Severity Issues

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 Severity Issues

Low-level severity issues can cause minor impact and 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 four 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.

A. Contract - FoundCoin.sol

High Severity Issues

A.1 Incorrect Decimal Handling in Token Minting function **mintToken**

Line	Function - mintToken
150	function mintToken(uint256 addSupply) public onlyOwner {
	_mint(msg.sender, addSupply);
	}

Description

The **mintToken** function incorrectly mints tokens without considering the token's decimal places. This can lead to a significant disparity between the intended and actual amount of tokens minted, potentially causing economic imbalances within the token ecosystem.

Remediation

1. **Adjust Decimal Handling:** Modify the **mintToken** function to include the token's decimal places when minting new tokens. This ensures that the minted amount accurately reflects the intended supply increase.
2. **Code Modification:** The corrected code snippet should multiply the **addSupply** parameter by **10^18** to account for the standard 18 decimal places used in ERC-20 tokens. If the token uses a different decimal system, adjust the multiplier accordingly.

Status

Resolved

CCFound Team's Comment

The decision is to retain the current approach, where decimals are manually added during minting.

A.2 Owner can renounce while system is paused using renounceOwnership function

Line	Function - mintToken
7	import "@openzeppelin/contracts/access/Ownable.sol";

Description

The Owner of the contract is usually the account that deploys the contract. As a result, the Owner is able to perform some privileged functions like mintToken(), includeInFees(), pause(), unpause() and blacklist() etc. In the FoundCoin.sol smart contract, the renounceOwnership function is used to renounce the Owner permission. Renouncing ownership before transferring would result in the contract having no Owner, eliminating the ability to call privileged functions.

Remediation

It is recommended that the Owner is not able to call renounceOwnership without transferring the Ownership to another address before. In addition, if a multi-signature wallet is used, calling renounceOwnership function should be confirmed for two or more users. As another solution, Renounce Ownership functionality can be disabled.

Status

Resolved

CCFound Team's Comment

The trust in the owner’s responsibility and the belief that the owner will not commit errors that could negatively impact the system.

Medium Severity Issues

A.3 Centralization Risk for trusted owners

Line	Functions - _mintNFTAfterDeposit
83	function updateRevShareWallet(address newRevShareWallet) external onlyOwner {
97	function withdrawStuckToken() external onlyOwner {
103	function withdrawStuckToken(address _token, address _to) external onlyOwner {
109	function withdrawStuckEth(address toAddr) external onlyOwner {
117	function blacklist(address _addr) public onlyOwner {
122	function unblacklist(address _addr) public onlyOwner {
126	function pause() public onlyOwner {
130	function unpause() public onlyOwner {
134	function includeInFees(address _addr) public onlyOwner {
138	function excludeFromFees(address _addr) public onlyOwner {
142	function mintToken(uint256 addSupply) public onlyOwner {

Description

Contracts have owners with privileged rights to perform admin tasks and need to be trusted to not perform malicious updates or drain funds.

Remediation

Make the **owner** a multi-sig and/or introduce a timelock for improved community oversight.

Status

Resolved

CCFound Team's Comment

It is an intended feature of the FoundCoin smart contract design.

Low Severity Issues

A.4 Single-step ownership transfer

Line	Functions - _mintNFTAfterDeposit
7	import "@openzeppelin/contracts/access/Ownable.sol";

Description

Inheriting from OpenZeppelin's Ownable contract means you are using a single-step ownership transfer pattern. If an admin provides an incorrect address for the new owner this will result in none of the 'Operator' marked methods being callable again. The better way to do this is to use a two-step ownership transfer approach, where the new owner should first claim its new rights before they are transferred.

Remediation

Use OpenZeppelin's Ownable2Step instead of Ownable.

Status

Resolved

CCFound Team's Comment

The approach of handling it as a single step is a deliberate choice made by the team.

A.5 Unchecked Transfer Return Values

Line	Functions
99	IERC20(address(this)).transfer(msg.sender, balance);
106	IERC20(_token).transfer(_to, _contractBalance);

Description

The ERC-20 **transfer** function, according to the ERC-20 standard, returns a boolean value indicating the success or failure of the operation. Ignoring this return value can lead to situations where a transfer fails (due to reasons like insufficient balance, or transfer to a contract that doesn't accept tokens), but the contract continues execution as if it had succeeded. This oversight could cause discrepancies in the contract's internal accounting, leading to potential vulnerabilities or loss of funds under specific conditions.

Remediation

1. Using **require** to Assert Successful Transfer
2. Using SafeERC20 Library from OpenZeppelin: The OpenZeppelin contracts library provides a **SafeERC20** wrapper around the ERC20 interface, which automatically checks the return value of **transfer**, **transferFrom**, and other ERC-20 operations, and reverts the transaction if any operation fails.

Status

Resolved

A.6 State-changing methods are missing event emissions

Line	Functions - NA
NA	function updateFees(uint256 _revShareFee, uint256 _burnFee) external onlyOwner {} ....etc.

Description

Critical functions within the contract lack the emission of events. Events play a vital role in providing transparency, enabling external systems to react to changes, and offering a way to track important contract activities. The absence of events can hinder monitoring and auditing efforts, making it difficult to detect and respond to critical contract actions.

Remediation

To address this issue and improve the transparency and auditability of the contract, you should add appropriate event emissions in critical functions. These events should capture essential information about the function's execution, including input parameters and outcomes. Additionally, consider emitting events both before and after critical state changes, where applicable.

Status

Resolved

Informational Issues

A.7: Consider using custom errors

Description

Custom errors reduce the contract size and can provide easier integration with a protocol. Consider using those instead of require statements with string error.

Status

Resolved

Functional Tests

Some of the tests performed are mentioned below:

- Should initiate the contract with provided address
- Should initiate the contract with provided owner address
- Should Set the contract Variables by Owner
- Should withdraw all funds
- Should transfer funds to another account
- Should Set the contract Variables by Owner
- Should not deposit if not whitelisted
- Should revert deposit if token address is invalid

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 Found Coin. We performed our audit according to the procedure described above.

Some issues of High, Medium, Low and informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture. In the End of the Audit, All Issues are closed.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in Found Coin smart contracts. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of Found Coin smart contracts. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of the Found Coin to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

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.

850+
Audits Completed

$30B
Secured

$30B
Lines of Code Audited

Follow Our Journey

Audit Report
February, 2024

For

ccfound

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com