QuillAudits

Audit Report
July, 2023

For

Table of Content

Executive Summary

Project Name	Kichee
Overview	In contract users can stake tokens and earn rewards. There is no time constraint or anything for which users can stake tokens. That means they can withdraw at any time. The only thing important here is that if they want to withdraw early then they cannot as there is a time limit between unstaking and withdrawal.
Timeline	22 November, 2022 - 8 December, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyze Staking Contract codebase for quality, security, and correctness. *https://mumbai.polygonscan.com/address/0x3cdeb5cc436ed39ca53f7807ce7c07ee2d4a9bc9#code*
Fixed in	*https://polygonscan.com/address/0xecef5b1e1040679a85ca2cead36de3f533d63e79#code*

2
Issues Found

🟥 High 🟨 Medium

🟩 Low 🟪 Informational

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

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
- ✓ Exception Disorder
- ✓ Gasless Send
- ✓ Use of tx.origin
- ✓ Compiler version not fixed
- ✓ Address hardcoded
- ✓ Divide before multiply
- ✓ Integer overflow/underflow
- ✓ Dangerous strict equalities
- ✓ Tautology or contradiction
- ✓ Return values of low-level calls
- ✓ Missing Zero Address Validation
- ✓ Private modifier
- ✓ Revert/require functions
- ✓ Using block.timestamp
- ✓ Multiple Sends
- ✓ Using SHA3
- ✓ Using suicide
- ✓ Using throw
- ✓ Using inline assembly

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 - Staking

High Severity Issues

No issues found

Medium Severity Issues

A.1 Centralization issue in staking contract.

Description

In staking contract functions: setMinimum(), setMax(), setMultiplier(), setlockingTime(), SetPenaltyMultiplier(), setAPY(), setPenaltyPercentage(), setStakingHardCap(), setStakingLock() and withdrawTokensAdmin() are controlled by owner. More importantly losing private key in some situation can allow modifying and losing control over this important functionality and hacker can drain contract by calling withdrawTokensAdmin().

Remediation

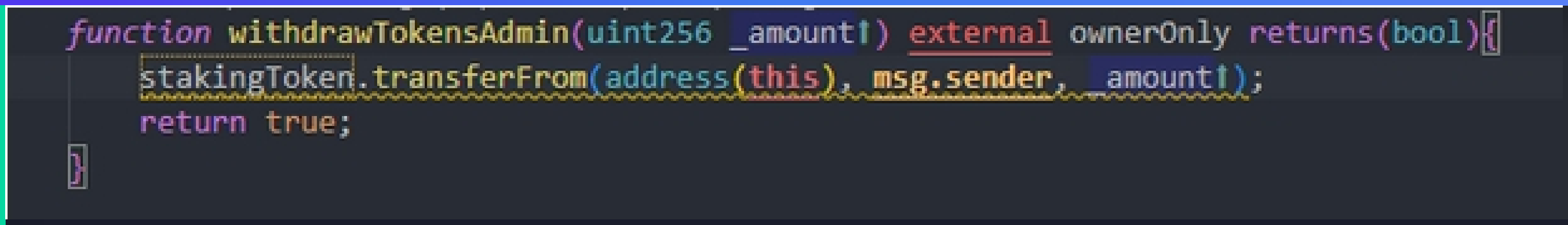
It is recommended to use multisig wallet such as gnosis safe.

Status

Acknowledged

Low Severity Issues

A.2 Use of transferFrom() for withdrawal from contract



```
function withdrawTokensAdmin(uint256 _amount) external ownerOnly returns(bool){
    stakingToken.transferFrom(address(this), msg.sender, _amount);
    return true;
}
```

Description

In staking contract transferFrom() is used to withdraw tokens by admin.

Remediation

When you transfer a token from smart contract to address you must use transfer() function instead transferFrom() because this function requires approve + transfer and smart contracts cannot approve itself. While transfer() is used to only transfer funds.

Status

Resolved

Informational Issues

No issues found

General Recommendation

- stakeId and totalStakedAmount can be kept without assigning values to 0 so that it saves gas.
- As stakingToken, owner is set in the constructor only it can be made immutable to save gas.

Status

Acknowledged

Functional Tests

Some of the tests performed are mentioned below

- ✓ Return values
- ✓ stakeTokens() (43ms)
- ✓ unstakeTokens()
- ✓ should check checkWalletBalance()
- ✓ should check checkCurrentByStakeId() (52ms)
- ✓ stake ID check (83ms)
- ✓ Check rewards (45ms)
- ✓ Withdraw from wallet() (82ms)
- ✓ should not be able to withdraw() before unstake (79ms)
- ✓ should not be able to withdraw() twice
- ✗ Should allow withdrawTokensAdmin()

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.

Summary

In this report, we have considered the security of the Staking Contract. 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.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the Kichee 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 Kichee 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.

850+
Audits Completed

$16B
Secured

800K
Lines of Code Audited

Follow Our Journey

Audit Report
July, 2023

For

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com