QuillAudits

Audit Report
February, 2023

For

DIGIHEALTH

Table of Content

Executive Summary

Project Name	Digihealth by Digipharm
Overview	Digihealth is a BEP20 contract with total supply of 100000000 and a blacklist functionality. Only the contract owner is permitted to blacklist address and this invariably prevents blacklisted address from carrying out any transfer of token or the spending of tokens. The token contract inherits the standard Openzeppelin contracts to enhance token operation. Mint functionality is restricted to the owner but ensured it does not exceed total supply.
Timeline	4 Feb, 2023 to 6 Feb, 2023
Method	Manual Review, Functional Testing, Automated Testing, etc.
Scope of Audit	The scope of this audit was to analyse Digihealth codebase for quality, security, and correctness. *https://bscscan.com/address/0xA87584Cfeb892C33A1C9a233e4A733b45c4160E6*

3
Issues Found

🟥 High 🟨 Medium
🟩 Low 🟪 Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	1	2
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	0	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
- ✓ Malicious libraries
- ✓ Compiler version not fixed
- ✓ Address hardcoded
- ✓ Divide before multiply
- ✓ Integer overflow/underflow
- ✓ ERC20 transfer() does not return boolean
- ✓ ERC20 approve() race
- ✓ 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
Mythril, Slither, SmartCheck,RemixIDE, Surya, Solhint.

Manual Testing

High Severity Issues

No issues were found

Medium Severity Issues

No issues were found

Low Severity Issues

A.1 Ownership Transfer must be a Two-step Processes

Description

Contracts are integrated with the standard Openzeppelin ownable contract, however, when the owner mistakenly transfers ownership to an incorrect address, ownership is completely removed from the original owner and cannot be reverted. The transferOwnership() function in the ownable contract allows the current owner to transfer his privileges to another address. However, inside transferOwnership() , the newOwner is directly stored in the storage, after validating the newOwner is a non-zero address, which may not be enough.

Remediation

It would be much safer if the transition is managed by implementing a two-step approach: _transferOwnership() and _updateOwnership() . Specifically, the _transferOwnership () function keeps the new address in the storage, _newOwner , instead of modifying the _owner() directly. The updateOwnership() function checks whether _newOwner is msg.sender, which means _newOwner signs the transaction and verifies himself as the new owner. After that, _newOwner could be set into _owner.

Status

Acknowledged

Informational Issues

A.2 Unlocked pragma (pragma solidity^0.8.10)

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. Moreover, we strongly suggest not to use experimental Solidity features (e.g., pragma experimental ABIEncoderV2) or third-party unaudited libraries. If necessary, refactor the current code base to only use stable features.

Status

Acknowledged

A.3 Absence of Code Comments

Remediation

Being a contract that modified the feature of some functions inherited from libraries, there was no code comments on these functions to explain the motive behind the modification. Code comments are relevant for contract users to comprehend contracts to an extent.

Remediation

Code comments added to contract are a better way to explain to users or those interacting with the contract, the underlying of these functions.

Status

Acknowledged

General Recommendation

DigiHealth token contract inherited a standard library that aids the creation of ERC20 token. The total supply is fixed to 100,000,000 and features the mint function that allows only the owner to mint tokens to addresses until it reaches the total supply. However, two functions in the token contract were modified; transfer and transferFrom functions - this is to disable blacklisted addresses from benefiting from these tokens. The modification poses issues of centralization and risks of losing tokens in these blacklisted addresses.

As per Digipharm White paper, it states the benefits that surround owning some DGH tokens and also cites its awareness of losing tokens to blacklisted addresses so as to prevent these addresses from earning from its economic value.

Functional Testing

Some of the tests performed are mentioned below:

- ✓ Blacklisted User should not be able to Transfer
- ✓ OnlyOwner can Blacklist any Account
- ✓ Should not be able to transfer more than Total supply
- ✓ Should be able to renounce ownership
- ✓ Only owner can RenounceOwnership
- ✓ Should prevent blacklisted addresses from spending allowed tokens.

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 Digihealth. We performed our audit according to the procedure described above.

Some issues of 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 Digihealth 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 Digihealth 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.

700+
Audits Completed

$16B
Secured

700K
Lines of Code Audited

Follow Our Journey

Audit Report
February, 2023

For

DIGIHEALTH

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com