QuillAudits

Audit Report
August, 2022

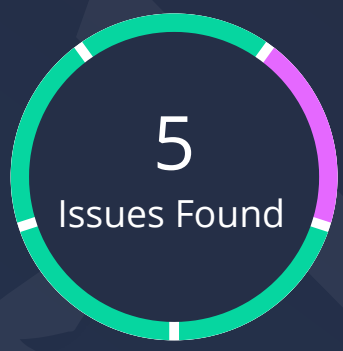
For

Dfyn

Table of Content

Executive Summary

Project Name	DFYN-RFQ
Overview	Dfyn Performs Swap from token0 to token1, based on the quote received from the marketMaker. Quotes are signed data, so they cannot be changed.
Timeline	20 July, 2022 to 9th August, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyse Dfyn RFQ codebase for quality, security, and correctness. *https://github.com/dfyn/dfyn-RFQ/commit/50321ac83ac649a84da4fa353a56a3da3d68d169*
Commit Hash	50321ac83ac649a84da4fa353a56a3da3d68d169
Review 2	*https://github.com/dfyn/dfyn-RFQ/commit/3e9117f04242f7b0d95aa5675a0a6ec78f824288*



High Medium

Low Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	4	1
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
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- BEP20 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 level

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: dfyn-RFQ

High Severity Issues

No issues were found

Medium Severity Issues

No issues were found

Low Severity Issues

A.1 Zero check missing

Description

The constructor lacks a zero check.

Remediation

We should put a require check in the constructor to check the address of _weth.

Status

Acknowledged

A.2 Owner should be multisig

Description

We recommend using a multisig account address (gnosis-safe) for the owner such that the only owner functions are not malicious in future and the decentralisation is achieved in the system.

Status

Acknowledged

A.3 Pragma is not locked

Description

The pragma versions used in the contract are not locked. Consider using the latest versions among 0.8.10 for deploying the contracts and libraries as it does not compile for any other version and can be confusing for a developer. Solidity source files indicate the versions of the compiler they can be compiled with.

pragma solidity ^0.8.0; // bad: compiles between 0.8.0 and 0.8.13
pragma solidity 0.8.0; // good : compiles w 0.8.0 only but not the latest version
pragma solidity 0.8.13; // best: compiles w 0.8.13

Remediation

Fix the solidity version by removing the caret symbol and use the best compiles method(specified version numbers).

Status

Acknowledged

A.4 No rescueFund method is implemented in contract

Recommendation

There should be a rescueFund method to rescue the tokens funds which are mistakenly sent and are not part of the contracts.

The behaviour of the rescueFund should only be operated by the owner and nobody else.

Status

Acknowledged

Informational Issues

A5. Recommendations

Description

- For test stake use *smock* Library.
- The naming convention (_swapFee) of updateTokenFee is bad.
- Gas optimization
  - _calculateFee should remove the safeMath wrapper in div.
- updateFeeAddress, addSigner, updateTokenFee have bad error message.

Status

Acknowledged

Functional Testing (Goërli)

Contracts
dyfn- 0xB8bbF3B3f1884DDF0fE7E7b269c2422a2bb995A1

Transactions
Add white list
https://goerli.etherscan.io/
tx/0xa9d1f141df74a6bdbe97cccdb0641652a92fb084b6b37182980c2d9867f73555

Add signer -
https://goerli.etherscan.io/
tx/0x8e4dedff903ca92d13399ffa6877de7a3e0bf82f7823837d6aabdb8da65d1b8b

Approve 5 eth -
https://goerli.etherscan.io/
tx/0x4d65c78071f8a209d38a91b9b920c5cc94178367866b4c89395229166c832753

Approve 5 PT_QH (mock ERC20) token -
https://goerli.etherscan.io/
tx/0x76efaa63d48fbcc13077b354624acde45ab03072c5019e126151a78ef92fdf4f

swapNativeToToken -
https://goerli.etherscan.io/
tx/0xf59201528bd809ff9c30776686e405c27b6d10fcc51222edde54aaf515016afd

swapTokens -
https://goerli.etherscan.io/
tx/0x795d3419d111fc212d14162b1a3ed05d1d4258bcd044c9dbab0c5272bc52ac46

swapTokenToNative -
https://goerli.etherscan.io/
tx/0x244e0cada3368cd8d47e9d8fd901e2ef2468d967f9a41f74e4b715e25b94ffb8

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 DFYN-RFQ. We performed our audit according to the procedure described above.Overall, smart contracts are well written and adhere to guidelines.

Some issues were discovered in the initial audit and the DFYN Team Acknowledged all issues.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the DFYN 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 DFYN 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
August, 2022

For

Dfyn

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com