QuillAudits

Audit Report
June, 2021

Contents

Scope of Audit

The scope of this audit was to analyze and document the Dare Token smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- 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 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 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.
SmartCheck.

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 security of smart contracts.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerability 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 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 possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Slither, SmartCheck.

Issue Categories

Every issue in this report has been assigned with 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 to be fixed before moving to a live environment.

Medium level 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 level severity issues

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 four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	1	1
Closed	0	0	0	0

Introduction

During the period of **June 16, 2021 to June 17, 2021** - QuillAudits Team performed a security audit for Dare Token smart contracts.

The code for the audit was taken from following the official link: https://bscscan.com/token/0xd589699b20903fc061701ad6a35cb78bb5dd734e

Issues Found – Code Review / Manual Testing

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low level severity issues

1. **Unchecked return value of ERC20 transfer() call**
   Description:
   The return value of an external ERC20 transfer() call has not been checked. It should return true for correct ERC-20 implementation.

   Remediation:
   Use OpenZeppelin's SafeERC20 implementation or ensure that the transfer() return value is checked.

   Status: Open

Informational

2. Different pragma directives are used

Version used:

- 0.8.3
- ^0.8.0

Description:
It is detected when different Solidity versions are used.

Remediation:
Use one Solidity version for all contracts. The pragma version 0.8.0 is safe to use in this token.

Status: Open

Functional test

Function Names	Testing results
name	Passed
decimals	Passed
balanceOf	Passed
allowance	Passed
totalSupply	Passed
symbol	Passed
allowance	Passed
transfer	Passed
approve	Passed
transferFrom	Passed
renounceOwnership	Passed
owner	Passed
increaseAllowance	Passed
decreaseAllowance	Passed
burn	Passed
returnAccidentalERC20	Passed

Automated Testing

Slither

```
INFO:Detectors:
DareToken.returnAccidentalERC20(address,address,uint256) (DareToken.sol#20-30) ignores return value by IERC20(_token).transfer(_to,_amount) (DareToken.sol#26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
DareToken.constructor(uint256)._totalSupply (DareToken.sol#10) shadows:
        - ERC20._totalSupply (@openzeppelin\contracts\token\ERC20\ERC20.sol#38) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Reentrancy in DareToken.returnAccidentalERC20(address,address,uint256) (DareToken.sol#20-30):
        External calls:
        - IERC20(_token).transfer(_to,_amount) (DareToken.sol#26)
        Event emitted after the call(s):
        - ReturnedERC20(_token,_to,_amount) (DareToken.sol#28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.8.3', '>=0.4.22<0.9.0', '^0.8.0']
        - ^0.8.0 (@openzeppelin\contracts\utils\Context.sol#3)
        - 0.8.3 (DareToken.sol#1)
        - ^0.8.0 (@openzeppelin\contracts\token\ERC20\ERC20.sol#3)
        - ^0.8.0 (@openzeppelin\contracts\token\ERC20\IERC20.sol#3)
        - ^0.8.0 (@openzeppelin\contracts\token\ERC20\extensions\IERC20Metadata.sol#3)
        - >=0.4.22<0.9.0 (Migrations.sol#2)
        - ^0.8.0 (@openzeppelin\contracts\access\Ownable.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Pragma version^0.8.0 (@openzeppelin\contracts\utils\Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.3 (DareToken.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC20\ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC20\IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC20\extensions\IERC20Metadata.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/
.7.6
Pragma version>=0.4.22<0.9.0 (Migrations.sol#2) is too complex
Pragma version^0.8.0 (@openzeppelin\contracts\access\Ownable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.3 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter DareToken.burn(uint256)._amount (DareToken.sol#14) is not in mixedCase
Parameter DareToken.returnAccidentalERC20(address,address,uint256)._token (DareToken.sol#20) is not in mixedCase
Parameter DareToken.returnAccidentalERC20(address,address,uint256)._to (DareToken.sol#20) is not in mixedCase
Parameter DareToken.returnAccidentalERC20(address,address,uint256)._to (DareToken.sol#20) is not in mixedCase
Parameter DareToken.returnAccidentalERC20(address,address,uint256)._amount (DareToken.sol#20) is not in mixedCase
Variable Migrations.last_completed_migration (Migrations.sol#6) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (@openzeppelin\contracts\utils\Context.sol#21)" inContext (@openzeppelin\contracts\utils\Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (@openzeppelin\contracts\token\ERC20\ERC20.sol#60-62)
symbol() should be declared external:
        - ERC20.symbol() (@openzeppelin\contracts\token\ERC20\ERC20.sol#68-70)
decimals() should be declared external:
        - ERC20.decimals() (@openzeppelin\contracts\token\ERC20\ERC20.sol#85-87)
totalSupply() should be declared external:
        - ERC20.totalSupply() (@openzeppelin\contracts\token\ERC20\ERC20.sol#92-94)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (@openzeppelin\contracts\token\ERC20\ERC20.sol#99-101)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#111-114)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (@openzeppelin\contracts\token\ERC20\ERC20.sol#119-121)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#130-133)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#148-156)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#170-173)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#189-195)
setCompleted(uint256) should be declared external:
        - Migrations.setCompleted(uint256) (Migrations.sol#16-18)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (@openzeppelin\contracts\access\Ownable.sol#54-57)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (@openzeppelin\contracts\access\Ownable.sol#63-67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:. analyzed (7 contracts with 75 detectors), 32 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the Dare 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 Dare Token Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines.

No instances of Re-entrancy or Back-Door Entry were found in the contract.

Numerous issues of various severity levels were discovered during the audit. It is recommended to kindly go through the above-mentioned details and fix the code accordingly.

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com