QuillAudits

Audit Report
December, 2020

Contents

Introduction

During the period of December 2nd, 2020 to December 3rd, 2020 - Quillhash Team performed a security audit for Difuse Token smart contract. The code for audit was taken from following the official smart contract link:

Deployed at:

https://ropsten.etherscan.io/address/0xE2566Fff00b9E2e77A892c0fE9C0f8688623B06e#contracts

About the Contract

Description

The DifuseToken.sol is an ERC20 token contract. It allows users to transfer tokens, approve other addresses as well as burn the tokens.

Imports

The contract imports SafeMath.sol

Usages

The DifuseTokens contract has the following usages:

- It uses SafeMath for all uint256 data types

Functions

- transfer (address _to, uint256 _value)
- approve (address _spender, uint256 _value)
- transferFrom (address _from, address _to, uint256 _value)
- burn (uint256 _value)
- burnFrom (address _from, uint256 _value)

State Variables and Mappings

- name
- symbol
- decimals
- totalSupply
- mapping(address => uint256) public balanceOf;
- mapping(address => mapping(address => uint256)) public allowance;

Events

event Transfer(address indexed _from, address indexed _to, uint256 _value)
event Approval(address indexed _owner, address indexed _spender, uint256 _value)

Scope of Audit

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

Check 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
- Implicit visibility level
- 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

Techniques and Methods

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

- Overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behavior.
- Token distribution and calculations are as per intended behavior mentioned in 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

	High	Medium	Low	Informational
Open	0	0	0	3
Closed	0	0	4	0

Issues Found – Code Review / Manual Testing

High severity issues

None.

Medium severity issues

None.

Low level severity issues

1. **External Visibility should be preferred**
   Those functions that are never called throughout the contract should be marked as external visibility instead of public visibility.
   This will effectively result in Gas Optimization as well.
   Therefore, the following function must be marked as external within the contract:

transfer	#Line 37
approve	#Line 46
transferFrom	#Line 52
burn	#Line 63
burnFrom	#Line 71

   Status: CLOSED

2. **Checking for ZERO Addresses**
   Difuse Token contract doesn't check for Zero Addresses before executing some of its most imperative functions.
   The following functions must check whether or not the address passed in the arguments are valid and revert back if the address is a Zero Address:

transfer	#Line37
transferFrom	#Line52
approve	#Line 46

   Status: CLOSED

3. **Require Statements do not include the Error Messages**
   None of the require statements in the Difuse Token contract include error messages.
   Including the error messages within the requirements enhances the code readability.
   Therefore, the error messages must be added.
   Status: CLOSED

4. **Compiler version should be fixed**
   Solidity source files indicate the versions of the compiler they can be compiled with. It's recommended to lock the compiler version in code, as future compiler versions may handle certain language constructions in a way the developer did not foresee. Also, it's recommended to use the latest compiler version.
   Status: CLOSED

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.

1. **Difuse Token contract doesn't include Pausable Functionalities**
   A pausable contract shall provide some additional layers of security to the token contract in case of emergencies.
   The current token contract doesn't include the Pausable functionalities.
   Status: OPEN

2. **Difuse Token contract doesn't include Owner and Ownable Functionalities.**
   The current token contract simply transfers the minted tokens to the address that deploys the contract.
   While there is nothing wrong with it, a comparatively effective procedure would have been to include the Ownable contract and transfer the minted tokens to the owner's address.
   Implementing an Ownable contract also provides access to some of the crucial functionalities like transfer of Ownership as well as onlyOwner modifier etc.
   Status: OPEN

3. **Coding Style Issues**
   Coding style issues influence code readability and in some cases may lead to bugs in future. Smart Contracts have naming convention, indentation and code layout issues. It's recommended to use the Solidity Style Guide to fix all the issues.
   Consider following the Solidity guidelines on formatting the code and commenting for all the files. It can improve the overall code quality and readability.
   Status: OPEN

Closing Summary

Overall, the smart contracts are very well written and adhere to guidelines. All the low severity issues that were found during the first audit have now been fixed. The informational issues that were found during the first audit have not been fixed. There were no critical or major issues found that can break the intended behavior

Disclaimer

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

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

hello@quillhash.com