QuillAudits

Audit Report
February, 2022

For

AQEX
AQUARIUS EXCHANGE

Contents

Scope of the Audit

The scope of this audit was to analyse and document the AQEX 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 maths
- 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.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
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.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	1	0	1
Closed	1	0	2	2

Introduction

During the period of Feb 7, 2022 to Feb 9, 2022 - QuillAudits Team performed a security audit for AQEX token smart contracts.

The code for the audit was taken from following the official link (Flattened_Token):

Codebase: 3628abfa6ea2636866f7d5ad1f8cc570b0d5bb1b
Fixed In: b9279a4407f44087d8be5c8d2d20aec2626b92ca

Fixed code is deployed on ropsten testnet by Aqex team: https://ropsten.etherscan.io/address/0x2bd8f83a48b623c5dd4caa83ee37fe7a755e94b5#code

Issues Found – Code Review / Manual Testing

High severity issues

1. Burn rights given to onlyOwner

The burn rights is provided to onlyOwner which signifies that owner can burn any user token. The behaviour is centralised in nature and not fully acceptable by the blockchain community.

Remediation
We recommend to use the implementation of burn to a specific user and anyone can call and burn his own token.

```
function burn(uint256 amount) external {
    _burn(_msgSender(), amount);
}
function _burn(address account, uint256 amount) internal {
    require(account != address(0), "ERC20: burn from the zero address");
    _beforeTokenTransfer(account, address(0), amount);
    uint256 accountBalance = _balances[account];
    require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
    unchecked {
        _balances[account] = accountBalance - amount;
    }
    _totalSupply -= amount;
    emit Transfer(account, address(0), amount);
    _afterTokenTransfer(account, address(0), amount);
}
```

Status: Fixed

Medium severity issues

2. setter for _rem_supply is missing

If the owner wan't to increase the _rem_supply in future then as per current the implementation owner can't do that. We recommend the user to make a setter for the _rem_supply with onlyOwner modifier.

Status: Acknowledged

Low severity issues

3. Variable visibility not defined

On line 12 the visibility of _rem_supply has not been defined. We recommend using private visibility.

Status: Fixed

4. Used locked pragma version

The pragma versions used in the contract are not locked. Consider using the latest versions among 0.8.11 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 0.8.11
pragma solidity 0.8.0; // good : compiles w 0.8.0 only but not the latest version
pragma solidity 0.8.11; // best: compiles w 0.8.11

Status: Fixed

Informational issues

5. Missing comments and description

Comments and Description of the methods and the variables are missing, it's hard to read and understand the purpose of the variables and the methods in context of the whole picture

Recommendation
Consider adding NatSpec format comments for the comments and state variables

Status: Acknowledged

6. **Public methods only being used externally**

   ‘public’ functions that are never used within the contract should be declared ‘external’ to save gas

   Recommendation
   Make these methods external
   name, symbol, decimal, totalSupply, rem_supply, balanceOf, allowance, approve, transferFrom, increaseAllowance, decreaseAllowance, _mint, transferOwnership, renounceOwnership

   Status: Fixed

7. **_mint name should changed to mint**

   Mostly _mint is used for internal methods, not for external/public. We recommend using mint instead of _mint.

   Status: Fixed

Kovan Testnet Test Contract

AQEX Token: 0xe05B70714CC6a0978FDAd250E022481376544404C

- _burn
  - SuccessFully burn token for other user PASS
  - Only Owner can burn the tokens PASS

- _mint
  - SuccessFully mint tokens to other user PASS
  - Only Owner can mint the tokens PASS

- approve
  - SuccessFully approve tokens to other spender PASS

- decreaseAllowance
  - decrease allowance by 5 PASS
  - greater than current Allowance PASS

- increaseAllowance
  - SuccessFully increase the allowance of spender PASS

- transfer
  - transfer 10 tokens to other user PASS
  - Not allowed to transfer to ZERO address PASS

- transferFrom
  - SuccessFully transfer from the approved allowance PASS
  - can't send more than the allowance PASS

- transferOwnership
  - SuccessFully transfer the ownership PASS
  - Only admin can transfer the ownership PASS

Automated Tests

Slither

```
ethsec@da5d0cdb0abf:/code/erc-20-token$ slither Flattened_Token.sol

FinalToken.allowance(address,address).owner (Flattened_Token.sol#100) shadows:
        - Ownable.owner() (Flattened_Token.sol#43-45) (function)
FinalToken._approve(address,address,uint256).owner (Flattened_Token.sol#174) shadows:
        - Ownable.owner() (Flattened_Token.sol#43-45) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Context._msgData() (Flattened_Token.sol#31-33) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (Flattened_Token.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Flattened_Token.sol#19) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Flattened_Token.sol#26) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Flattened_Token.sol#36) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Flattened_Token.sol#64) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function FinalToken.rem_supply() (Flattened_Token.sol#90-92) is not in mixedCase
Function FinalToken._mint(address,uint256) (Flattened_Token.sol#151-160) is not in mixedCase
Function FinalToken._burn(address,uint256) (Flattened_Token.sol#161-172) is not in mixedCase
Variable FinalToken._rem_supply (Flattened_Token.sol#69) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

FinalToken.slitherConstructorVariables() (Flattened_Token.sol#66-193) uses literals with too many digits:
        - _rem_supply = 500000000000000000000000000 (Flattened_Token.sol#69)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Flattened_Token.sol#50-52)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Flattened_Token.sol#53-56)
name() should be declared external:
        - FinalToken.name() (Flattened_Token.sol#78-80)
symbol() should be declared external:
        - FinalToken.symbol() (Flattened_Token.sol#81-83)
decimals() should be declared external:
        - FinalToken.decimals() (Flattened_Token.sol#84-86)
totalSupply() should be declared external:
        - FinalToken.totalSupply() (Flattened_Token.sol#87-89)
rem_supply() should be declared external:
        - FinalToken.rem_supply() (Flattened_Token.sol#90-92)
balanceOf(address) should be declared external:
        - FinalToken.balanceOf(address) (Flattened_Token.sol#93-95)
transfer(address,uint256) should be declared external:
        - FinalToken.transfer(address,uint256) (Flattened_Token.sol#96-99)
allowance(address,address) should be declared external:
        - FinalToken.allowance(address,address) (Flattened_Token.sol#100-102)
approve(address,uint256) should be declared external:
        - FinalToken.approve(address,uint256) (Flattened_Token.sol#103-106)
transferFrom(address,address,uint256) should be declared external:
        - FinalToken.transferFrom(address,address,uint256) (Flattened_Token.sol#107-121)
increaseAllowance(address,uint256) should be declared external:
        - FinalToken.increaseAllowance(address,uint256) (Flattened_Token.sol#122-125)
decreaseAllowance(address,uint256) should be declared external:
        - FinalToken.decreaseAllowance(address,uint256) (Flattened_Token.sol#126-133)
_mint(address,uint256) should be declared external:
        - FinalToken._mint(address,uint256) (Flattened_Token.sol#151-160)
_burn(address,uint256) should be declared external:
        - FinalToken._burn(address,uint256) (Flattened_Token.sol#161-172)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
Flattened_Token.sol analyzed (5 contracts with 75 detectors), 30 result(s) found
ethsec@da5d0cdb0abf:/code/erc-20-token$
```

Results

No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorised above according to their level of severity.

Closing Summary

Several issues of High, Medium, Low severity and information issues have been reported during the initial Audit ,which has been fixed by the Aqex team.
The contracts are good to deploy on public EVM chains.

Disclaimer

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

Audit Report
February, 2022

For

AQEX
AQUARIUS EXCHANGE

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com