QuillAudits

Audit Report
July, 2021

AGRIUT

Contents

Scope of Audit

The scope of this audit was to analyze and document the AgriUT 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	0	0
Acknowledged	0	1	0	0
Closed	0	0	0	4

Introduction

During the period of **July 15, 2021 to July 15, 2021** - QuillAudits Team performed a security audit for AgriUT smart contracts.

The code for the audit was taken from the following official link: https://github.com/Agreum/AgriUT/blob/main/AgriUTToken.sol

Note	Date	Commit hash
Version 1	July	2b175e68240e67cfebe30272cb374555f2a460fb
Version 2	July	a3638774be81be9f113ae0ddc54f4a5b84038947

Issues Found – Code Review / Manual Testing

High severity issues

No issues were found

Medium severity issues

1. **Description**
   The role owner has the authority to:
   - transfer user tokens
   - freeze user accounts
   - burn user tokens

   Remediation
   We advise the client to handle the governance account carefully to avoid any potential hack. We also advise the client to consider the following solutions:

   1. Time-lock with reasonable latency for community awareness on privileged operations;
   2. DAO or Governance module increasing transparency and community involvement;

   Status: Acknowledged by the Auditee
   Comments from Auditee: “The ability to interact with users’ wallets is a core part of the functionality we need to provide. To deal with the issue of governance the 3 functions mentioned all check for managedAccounts is true, which has to be set first, from the user wallet. This means we can’t pick a random account and perform the operations. The idea is that for all accounts which are part of the Agrigata network, we’ll fully manage & register them with approveOwnerToManage function. For users setting up their own wallets, they would not register their wallets, so we would be blocked from performing those operations.”

Low level severity issues

No issues were found

Informational

1. Absence of error messages in require statements

Description
While error messages in require() statements are optional, adding them helps provide description about the error faced by the user.

Remediation
We recommend adding error messages to the require statements.

Status: Closed
In version 2, require() statements were updated with error messages.

2. Missing Events for Significant Transactions

Description
The missing event makes it difficult to track off-chain liquidity fee changes. An event should be emitted for significant transactions calling the function :

- transferOwnership
- burnFrom

In the burnFrom() function, allowance is updated for the _from address, but no event is emitted for this change.

Remediation
We recommend emitting events for the above transactions.

Status: Closed
In version 2, appropriate events were emitted for the transactions.

3. Floating Pragma

pragma solidity ^0.8.5;

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
Lock the pragma version and also consider known bugs for the compiler version that is chosen.

Status: Closed
The Pragma version was locked to 0.8.6 in version 2.

4. Public function that could be declared external

Description
The following public functions that are never called by the contract should be declared external to save gas:

- burn()
- burnFrom()
- burnWithApproval()
- freezeAccount()
- approveOwnerToManage()
- transferFromGivenApproval()

Remediation
Use the external attribute for functions never called from the contract.

Status: Closed
In version 2, the above-mentioned functions were declared as external.

Functional test

Function Names	Testing results
name()	Passed
symbol()	Passed
decimals()	Passed
totalSupply()	Passed
balanceOf()	Passed
frozenAccount()	Passed
managedAccount()	Passed
transfer()	Passed
transferFrom()	Passed
transferFromGivenApproval()	Passed
approve()	Passed
approveOwnerToManage()	Passed
allowance()	Passed
freezeAccount()	Passed
burn()	Passed
burnWithApproval()	Passed
burnFrom()	Passed

Automated Testing

Slither

```
INFO:Detectors:
AgriUTToken._approve(address,address,uint256).owner (AgriUTToken.sol#153) shadows:
        - OwnableToken.owner() (OwnableToken.sol#12-14) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['=0.8.6', '>=0.4.22<0.9.0']
        - =0.8.6 (AgriUTToken.sol#2)
        - =0.8.6 (ERC20Interface.sol#2)
        - >=0.4.22<0.9.0 (Migrations.sol#2)
        - =0.8.6 (OwnableToken.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Pragma version=0.8.6 (AgriUTToken.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version=0.8.6 (ERC20Interface.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.4.22<0.9.0 (Migrations.sol#2) is too complex
Pragma version=0.8.6 (OwnableToken.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.6 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter AgriUTToken.frozenAccount(address)._account (AgriUTToken.sol#62) is not in mixedCase
Parameter AgriUTToken.managedAccount(address)._account (AgriUTToken.sol#68) is not in mixedCase
```

```
Parameter AgriUTToken.transfer(address,uint256)._to (AgriUTToken.sol#93) is not in mixedCase
Parameter AgriUTToken.transfer(address,uint256)._value (AgriUTToken.sol#93) is not in mixedCase
Parameter AgriUTToken.burn(uint256)._value (AgriUTToken.sol#186) is not in mixedCase
Parameter AgriUTToken.burnWithApproval(address,uint256)._from (AgriUTToken.sol#198) is not in mixedCase
Parameter AgriUTToken.burnWithApproval(address,uint256)._value (AgriUTToken.sol#198) is not in mixedCase
Parameter AgriUTToken.burnFrom(address,uint256)._from (AgriUTToken.sol#211) is not in mixedCase
Parameter AgriUTToken.burnFrom(address,uint256)._value (AgriUTToken.sol#211) 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:
AgriUTToken._decimals (AgriUTToken.sol#10) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
decimals() should be declared external:
        - AgriUTToken.decimals() (AgriUTToken.sol#32-35)
totalSupply() should be declared external:
        - AgriUTToken.totalSupply() (AgriUTToken.sol#38-41)
        - ERC20Interface.totalSupply() (ERC20Interface.sol#10)
name() should be declared external:
        - AgriUTToken.name() (AgriUTToken.sol#44-47)
symbol() should be declared external:
        - AgriUTToken.symbol() (AgriUTToken.sol#50-53)
```

```
managedAccount(address) should be declared external:
        - AgriUTToken.managedAccount(address) (AgriUTToken.sol#68-71)
transfer(address,uint256) should be declared external:
        - AgriUTToken.transfer(address,uint256) (AgriUTToken.sol#93-97)
        - ERC20Interface.transfer(address,uint256) (ERC20Interface.sol#13)
transferFrom(address,address,uint256) should be declared external:
        - AgriUTToken.transferFrom(address,address,uint256) (AgriUTToken.sol#106-115)
        - ERC20Interface.transferFrom(address,address,uint256) (ERC20Interface.sol#15)
approve(address,uint256) should be declared external:
        - AgriUTToken.approve(address,uint256) (AgriUTToken.sol#147-151)
        - ERC20Interface.approve(address,uint256) (ERC20Interface.sol#14)
setCompleted(uint256) should be declared external:
        - Migrations.setCompleted(uint256) (Migrations.sol#16-18)
transferOwnership(address) should be declared external:
        - OwnableToken.transferOwnership(address) (OwnableToken.sol#26-31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:. analyzed (4 contracts with 75 detectors), 30 result(s) found
```

Results

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

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the AgriUT 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 AgriUT 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 Integer Overflow and Underflow vulnerabilities or Back-Door Entry were found in the contract.

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com