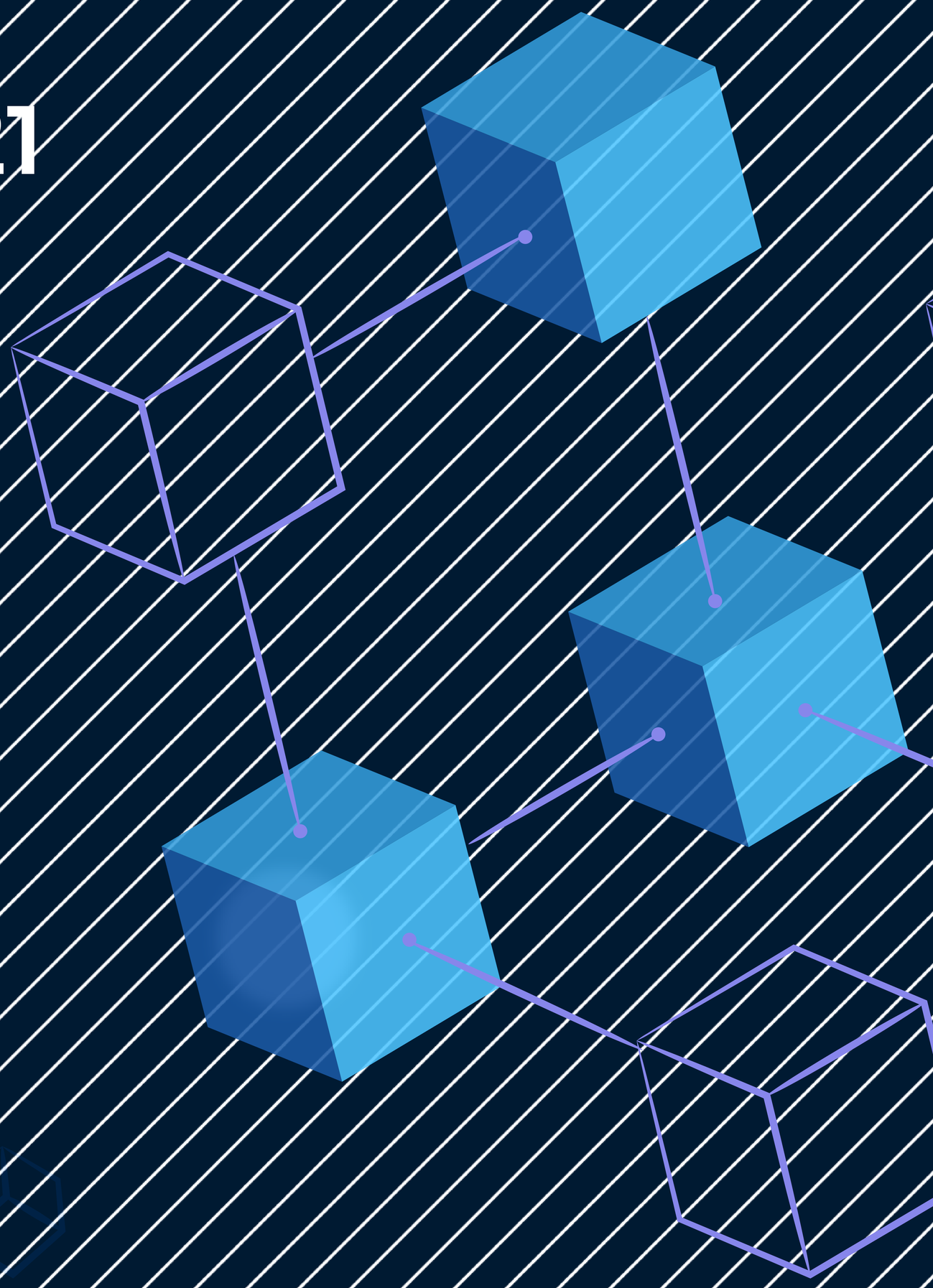
QuillAudits

Audit Report
December, 2021



For

Axis Token

Contents

Scope of the Audit

The scope of this audit was to analyze and document the Axis 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 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, Theo.

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	2
Closed	0	0	0	0

Introduction

During the period of **September 19, 2021 to September 20, 2021** - QuillAudits Team performed a security audit for **Axis Token** smart contracts.

The code for the audit was taken from following the official link:
https://etherscan.io/address/0xf0c5831ec3da15f3696b4dad8b21c7ce2f007f28#code

Issues Found

A. Contract – LineAxis

High severity issues

No issues were found.

Medium severity issues

1. **Centralization Risks**

   Description
   The role owner has the authority to :
   - **Mint** new tokens
   - **Freeze/Unfreeze** the transfers

   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:

   - Time-lock with reasonable latency for community awareness on privileged operations;
   - Multisig with community-voted 3rd-party independent co-signers;
   - DAO or Governance module increasing transparency and community involvement;

   Status: Acknowledged

Low severity issues

No issues were found.

Informational issues

2. 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:

- mint()
- freezeTransfers()
- unfreezeTransfers()
- transfer()
- transferFrom()

Remediation

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

Status: Acknowledged

3. Incorrect ERC20 implementation followed

Description

The functions like transfer and transferFrom do not return boolean values, which most of the tokens implement. So other contracts might face issues when interacting with this token contract.

Remediation

Set the appropriate return values and types for the defined ERC20 functions.

Status: Acknowledged

Functional test

Function Names	Testing results
name()	Passed
symbol()	Passed
decimals()	Passed
totalSupply()	Passed
balanceOf()	Passed
transfer()	Passed
approve()	Passed
allowance()	Passed
transferFrom()	Passed
mint()	Passed
freezeTransfers()	Passed
unfreezeTransfers	Passed
owner()	Passed
isOwner()	Passed
renounceOwner()	Passed
transferOwner()	Passed
cancelOwner()	Passed
confirmOwner()	Passed

Automated Tests

Slither

```
LineAxis (LineAxis.sol#268-328) has incorrect ERC20 function interface:ERC20.approve(address,uint256) (LineAxis.sol#229-234)
LineAxis (LineAxis.sol#268-328) has incorrect ERC20 function interface:ERC20.transferFrom(address,address,uint256) (LineAxis.sol#243-24
8)
LineAxis (LineAxis.sol#268-328) has incorrect ERC20 function interface:ERC20.transfer(address,uint256) (LineAxis.sol#262-264)
LineAxis (LineAxis.sol#268-328) has incorrect ERC20 function interface:LineAxis.transfer(address,uint256) (LineAxis.sol#319-321)
LineAxis (LineAxis.sol#268-328) has incorrect ERC20 function interface:LineAxis.transferFrom(address,address,uint256) (LineAxis.sol#324
-326)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

Different versions of Solidity is used:
        - Version used: ['>=0.4.22<0.9.0', '^0.4.24']
        - ^0.4.24 (LineAxis.sol#5)
        - >=0.4.22<0.9.0 (Migrations.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

ERC20CoreBase._burn(address,uint256) (LineAxis.sol#173-179) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.4.24 (LineAxis.sol#5) allows old versions
Pragma version>=0.4.22<0.9.0 (Migrations.sol#2) is too complex
solc-0.4.24 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable ERC20CoreBase._balanceOf (LineAxis.sol#105) is not in mixedCase
Variable ERC20CoreBase._totalSupply (LineAxis.sol#106) is not in mixedCase
Variable Migrations.last_completed_migration (Migrations.sol#6) is not in mixedCase


owner() should be declared external:
        - Owned.owner() (LineAxis.sol#37-39)
renounceOwner() should be declared external:
        - Owned.renounceOwner() (LineAxis.sol#62-65)
transferOwner(address) should be declared external:
        - Owned.transferOwner(address) (LineAxis.sol#71-74)
cancelOwner() should be declared external:
        - Owned.cancelOwner() (LineAxis.sol#77-79)
confirmOwner() should be declared external:
        - Owned.confirmOwner() (LineAxis.sol#81-85)
totalSupply() should be declared external:
        - ERC20CoreBase.totalSupply() (LineAxis.sol#119-121)
balanceOf(address) should be declared external:
        - ERC20CoreBase.balanceOf(address) (LineAxis.sol#129-131)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (LineAxis.sol#215-217)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (LineAxis.sol#229-234)
mint(address,uint256) should be declared external:
        - LineAxis.mint(address,uint256) (LineAxis.sol#293-295)
freezeTransfers() should be declared external:
        - LineAxis.freezeTransfers() (LineAxis.sol#301-306)
unfreezeTransfers() should be declared external:
        - LineAxis.unfreezeTransfers() (LineAxis.sol#312-317)
setCompleted(uint256) should be declared external:
```

Results

No major issues were found. Some false positive errors were reported by the tool.

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.
One Medium Issue found during the Audit, which has been Acknowledged by the Axis Team.

Disclaimer

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

Audit Report
December, 2021

For

Axis Token

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com