QuillAudits

Audit Report
February, 2023

For

Table of Content

Executive Summary

Project Name	Bitlearn
Overview	Bitlearn Galaxy is a Web 3.0 based decentralized education system "LEARN TO EARN" Platform.
Timeline	25 January, 2023 to 6 February, 2023
Method	Manual Review, Functional Testing, Automated Testing, etc.
Scope of Audit	The scope of this audit was to analyse Bitlearn codebase for quality, security, and correctness. *https://bscscan.com/token/0x251d2f9e346dd5a0806e47021f56242ed52c249d#code*
Fixed In	*https://bscscan.com/token/0x0f59f45f55f7b45b36f66a55baafd0f402729319#code*

🟥 High 🟨 Medium 🟩 Low 🟪 Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	**1**	**1**	0	**1**

Types of Severities

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.

Types of Issues

Open
Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved
These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged
Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved
Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- ✓ Re-entrancy
- ✓ Timestamp Dependence
- ✓ Gas Limit and Loops
- ✓ Exception Disorder
- ✓ Gasless Send
- ✓ Use of tx.origin
- ✓ Malicious libraries
- ✓ Compiler version not fixed
- ✓ Address hardcoded
- ✓ Divide before multiply
- ✓ Integer overflow/underflow
- ✓ ERC20 transfer() does not return boolean
- ✓ ERC20 approve() race
- ✓ Dangerous strict equalities
- ✓ Tautology or contradiction
- ✓ Return values of low-level calls
- ✓ Missing Zero Address Validation
- ✓ Private modifier
- ✓ Revert/require functions
- ✓ Using block.timestamp
- ✓ Multiple Sends
- ✓ Using SHA3
- ✓ Using suicide
- ✓ Using throw
- ✓ Using inline assembly

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

Mythril, Slither, SmartCheck,RemixIDE, Surya, Solhint.

Manual Testing

High Severity Issues

A.1 Token is not ERC20 compliant

Remediation

Token not being ERC20 compliant can be problematic and also can cause issues with many of the different protocols and projects.

Remediation

You can use openzeppelin libraries which have correct ERC20 compliance.

Status

Resolved

Medium Severity Issues

A.2 ERC20 approve is vulnerable to the front-running

Description

Consider:

1. Alice decided to approve bob 1000 tokens
2. But now for a reason she decides to let him spend 500 only
3. Bob sees the both txs in memepool and he calls transferFrom() on first txs which lets him spend 1000 tokens
4. As he enters high gas fees his transactions will be added before Alice's second approve of 500 tokens.
5. Now her second txs gets approved makes it Bob to spend 500 more tokens

Double spend attack issue

Remediation

It is recommended introducing increaseAllowance / decreaseAllowance functions. Also add approval of 0 tokens.

Status

Resolved

Low Severity Issues

No issues were found

Informational Issues

A.3 Unlocked pragma (pragma solidity ^0.8.2)

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

Here all the in-scope contracts have an unlocked pragma, it is recommended to lock the same. Moreover, we strongly suggest not to use experimental Solidity features (e.g., pragma experimental ABIEncoderV2) or third-party unaudited libraries. If necessary, refactor the current code base to only use stable features.

Status

Resolved

Functional Testing

Some of the tests performed are mentioned below:

- ✓ [PASS] testFail_transferFrom() (gas: 74918)
- ✓ [PASS] test_TransferBetweenTwoUsers() (gas: 74192)
- ✓ [PASS] test_approveFunction() (gas: 38357)
- ✓ [FAIL. Reason: EvmError: Revert Counterexample:
- ✓ calldata=0x08e21349000000000000000000000000000000000000000006765c793fa10079d0
- ✓ 000001, args=[2000000000000000000000000001]] test_balanceInvariant(uint256) (runs: 42,
- ✓ μ: 42248, ~: 42363)
- ✓ [PASS] test_balanceOf() (gas: 9932)
- ✓ [PASS] test_transferFrom() (gas: 81178)

Automated Tests

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

Closing Summary

In this report, we have considered the security of Bitlearn. We performed our audit according to the procedure described above.

Some issues of High, Medium, Low and informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

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

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

700+	**$16B**	**700K**
Audits Completed	Secured	Lines of Code Audited

Follow Our Journey

Audit Report
February, 2023

For

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com