QuillAudits

AUDIT REPORT

January, 2025

For

Beam

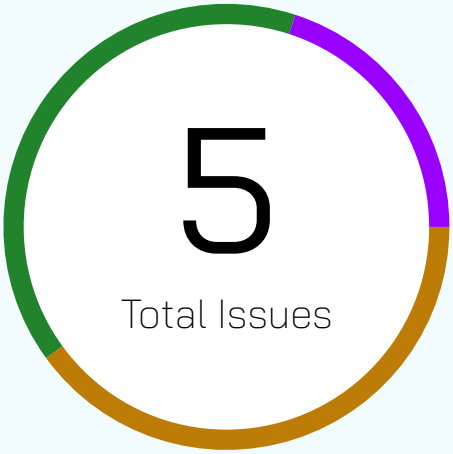
Table of Content

Executive Summary

Project name	Beam
Overview	The Beam protocol focuses on cross-chain DeFi applications, offering a robust ecosystem with integrated features for liquidity incentives, tokenomics, and governance.The protocol leverages ZetaChain for cross-chain messaging, enabling interactions between Bitcoin, Ethereum, and other ecosystems.
Audit Scope	The scope of this Audit was to analyze the Beam Smart Contracts for quality, security, and correctness. https://github.com/codemelt-dev/beam-contracts
Contracts in Scope	AlgebraSwapHelperLib.sol BitcoinMessageParserLib.sol MessageParserLib.sol SafeMath.sol UniswapV2Library.sol BitcoinRouter.sol MultichainRouter.sol
Commit Hash	2ce96319706e4e7acba9d4122820fbfa04a453cd
Language	Solidity
Blockchain	Zetachain
Method	Manual Analysis Functional Testing Automated Testing
Review 1	6th January 2025 - 15th January 2025
Updated Code Received	17th January 2025
Review 2	17th January 2025 - 22nd January 2025

Fixed In	3ae6a75d01e09ae595982ec57c242c17054d8d0e

Number of Issues per Severity



High	0 (0.00%)
Medium	2 (40.00%)
Low	2 (40.00%)
Informational	1 (20.00%)

Severity

Issues	High	Medium	Low	Informational
Open	0	0	0	0
Resolved	0	**2**	**2**	**1**
Acknowledged	0	0	0	0
Partially Resolved	0	0	0	0

Checked Vulnerabilities

Techniques and Methods

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

- The overall quality of code.
- Use of best practices.
- Code documentation and comments, match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.
- Implementation of ERC 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.

Static Analysis

A 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 analyzed, 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 behavior 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, Foundry, Solhint, Mythril, Slither, Solidity statistical analysis.

Types of Severity

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

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 be fixed before moving to a live environment.

Medium 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 Severity Issues

Low-level severity issues can cause minor impact and are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational Issues

These are four 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

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

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.

Medium Severity Issues

Router is not able to handle Fee on Transfer Tokens

Resolved

Path

MultichainRouter.sol, BitcoinRouter.sol

Function

_executeSwap

Description

The current implementation of _executeSwap might not be able to handle Fee on Transfer Tokens. This is because the router will receive less amount than expected and will then proceed to execute as if the input amount is same.

Recommendation

Either create a whitelisting mechanism to disallow these weird ERC20 tokens or handle them independently.

Implement a function to sweep dust accumulation

Resolved

Path

MultichainRouter.sol, BitcoinRouter.sol

Function

_executeSwap

Description

There is an edge case where rounding issue can create dust amount which will accumulate overtime whenever swap happens. However there is no way to pull those funds out.

Recommendation

Implement a sweep() function that is called by admin to sweep dust.

Low Severity Issues

Lack of reentrancy guard may result in unintended states

Resolved

Path

MultichainRouter.sol, BitcoinRouter.sol

Function

swapExactInput, swapExactInputSingle

Description

Currently swapExactInput , swapExactInputSingle lacks non-reentrant modifier. This means that there is a scenario under which one can reenter into these function before the previous execution is finished.
While an actual reentrant poc has not be crafted, it has been observed that most of exploits happen due to arbitrary user input. Therefore it is crucial to limit the user flexibility.

Recommendation

Add non-reentrant modifier and follow CEI pattern wherever necessary.

Consider sanitizing user inputs

Resolved

Path

MultichainRouter.sol, BitcoinRouter.sol

Description

The current code lack input sanitization of the parameters that are passed. This leads to break a whole lot of invariants that a router should hold creating in unintended consequences.

Here is a list of core invariants that should be checked.
A route must consist at least of two tokens.
RouterLogic must never consume more input tokens than the provided.
Only the Router can invoke swapExactInput/swapExactInputSingle.
Amounts must never be larger than uint128.max.
During swapExactInput, all input amounts must be consumed without any leftovers.
Token addresses must be unique.

Recommendation

Consider sanitizing these inputs against the invariants.

Informational Severity Issues

Unused Event, Error and INIT_CODE_HASH

Resolved

Description

Contract contains unused elements that do not serve any functional purpose, increasing the code-base size unnecessarily:

event: CustomApproval,
errors: WrongGasContract, NotEnoughToPayGasFee, ChainNotEnabled and
INIT_CODE_HASH: is declared but not utilized in any function

Recommendation

Remove unused event, error, and INIT_CODE_HASH

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 Beam. We performed our audit according to the procedure described above.

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

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in Beam. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of Beam. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of Beam to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

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

7+ Years of Expertise	**1M+** Lines of Code Audited
$30B+ Secured in Digital Assets	**1400+** Projects Secured

Follow Our Journey

AUDIT REPORT

January, 2025

For

Canada, India, Singapore, UAE, UK

www.quillaudits.com audits@quillaudits.com