QuillAudits

Audit Report, August, 2024

For

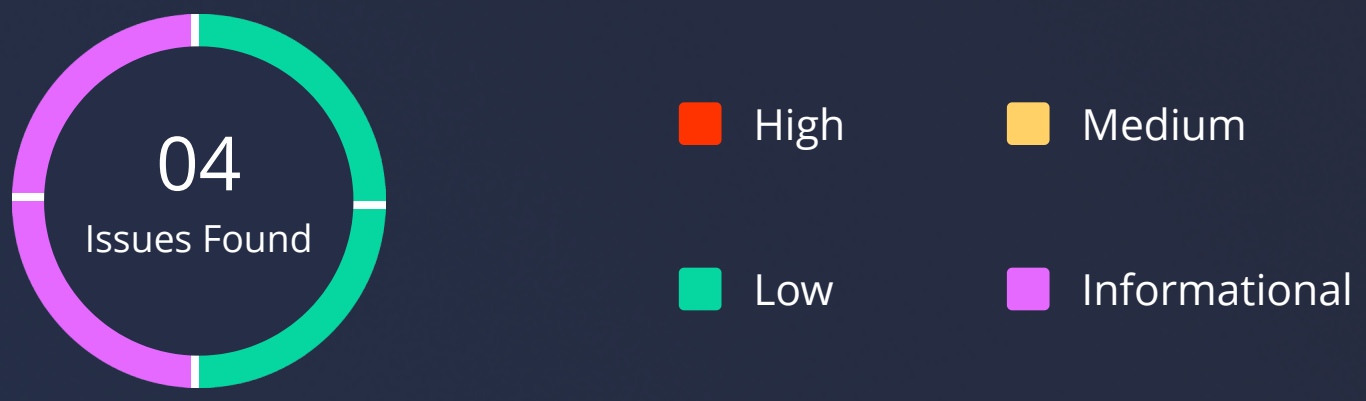
OMO Swap

Table of Content

Executive Summary

Project Name	OMOSwap
Project URL	https://app.omoswap.xyz/
Overview	Bridge.sol and CallProxy.sol are the contracts under scope of audit.These contracts form the core of a cross-chain bridge system for OmoSwap. The Bridge contract handles the main logic for sending and receiving tokens across different blockchain networks, while the CallProxy contract allows for executing arbitrary calls with the received tokens. Key points to note: 1. The system uses a token messenger contract for actual token transfers between chains. 2. There's a fee collection mechanism in place. 3. The bridge supports enabling/disabling specific tokens and routes. 4. The CallProxy allows for complex interactions on the receiving end of a bridge transfer. 5. Both contracts implement various security measures, including access control, reentrancy guards, and rescue functions.
Source Code	https://drive.google.com/drive/folders/1Z51t3Vpx2aauJln_q1HTorI-ghiMBkOj?usp=share_link
Audit Scope	The scope of the audit was to analyse code security and quality of OMOswap bridge contracts.
Contracts In-Scope	Bridge.sol and CallProxy.sol
Commit Hash	NA - Zip file was provide by OMOswap team
Language	Solidity
Blockchain	BSC
Method	Manual Review, Automated Tools.
Review 1	19th August 2024 - 26th August 2024

Number of Issues per Severity



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

Checked Vulnerabilities

- Access Management
- Arbitrary write to storage
- Centralization of control
- Ether theft
- Improper or missing events
- Logical issues and flaws
- Arithmetic Correctness
- Race conditions/front running
- SWC Registry
- 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
- ERC's conformance
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls
- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Multiple Sends
- Using suicide
- Using delegatecall
- Upgradeable safety
- Using throw

Checked Vulnerabilities

- Using inline assembly
- Style guide violation
- Unsafe type inference
- Implicit visibility level

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, Hardhat, Solhint, Slither, Mythril, 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

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

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.

A. Bridge and Call Proxy Contract

Low Severity Issues

A.1: Check for amount and recipient at source

Description

In the bridgeOut function, there is no check on amount and recipient address.These are validated in bridgeIn function at destination chain.

Recommendation

It is recommended to do these checks at source to avoid failures and fee loss of users.

Status

Acknowledged

A.2: Use custom errors

Description

The contracts use require statements for throwing errors.

Recommendation

It is recommended to use custom reverts for better readability on block chain explorers and save gas as strings take more gas as they occupy more bytes.

Status

Acknowledged

Informational Issues

A.3: Use SafeERC20 library

Description

As the contracts use ERC20 operations, it is recommended to use SafeERC20 lib for non standard ERC20 token transaction in the contract.

Status

Acknowledged

A.4: Consider adding tests

Description

No tests found for the contracts, please consider adding tests for the end to end testing of contracts and find out edge cases which might be missing during the implementation.

Status

Acknowledged

Functional Tests Cases

Some of the tests performed are mentioned below:

- [PASS] bridgeOut transfers amount from sender
- [PASS] bridgeOut checks the inputs
- [PASS] bridgeOut reverts if called by CallProxy
- [PASS] bridgeIn validates the attestations
- [PASS] bridgeIn transfers the token to recipient
- [PASS] bridgeIn args are properly deserialised
- [PASS] bridgeIn executes the calldata using call proxy
- [PASS] setters fail on calling from a non owner address

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 Bridge Smart Contract by OMOSwap. We performed our audit according to the procedure described above.

Some issues of Low and informational severity were found. In the End, OMOSwap team Acknowledged all issues.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in Bridge Smart Contract by OMOSwap. 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 Bridge Smart Contract by OMOSwap. 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 OMOSwap Team to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

About QuillAudits

QuillAudits is a leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With six years of expertise, we've secured over 1000 projects globally, averting over $30 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project's security with cutting-edge practices.

1000+
Audits Completed

$30B
Secured

1M+
Lines of Code Audited

Follow Our Journey

Audit Report
August, 2024

For

OMO Swap

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com