QuillAudits

Audit Report
July, 2024

For

Table of Content

Executive Summary 02

Number of Security Issues per Severity 03

Checked Vulnerabilities 04

Techniques and Methods 06

Types of Severity 07

Types of Issues 07

High Severity Issues 08

Medium Severity Issues 08

Low Severity Issues 08

Informational Issues 08

1. Improper use may result in funds loss 08

Automated Tests 10

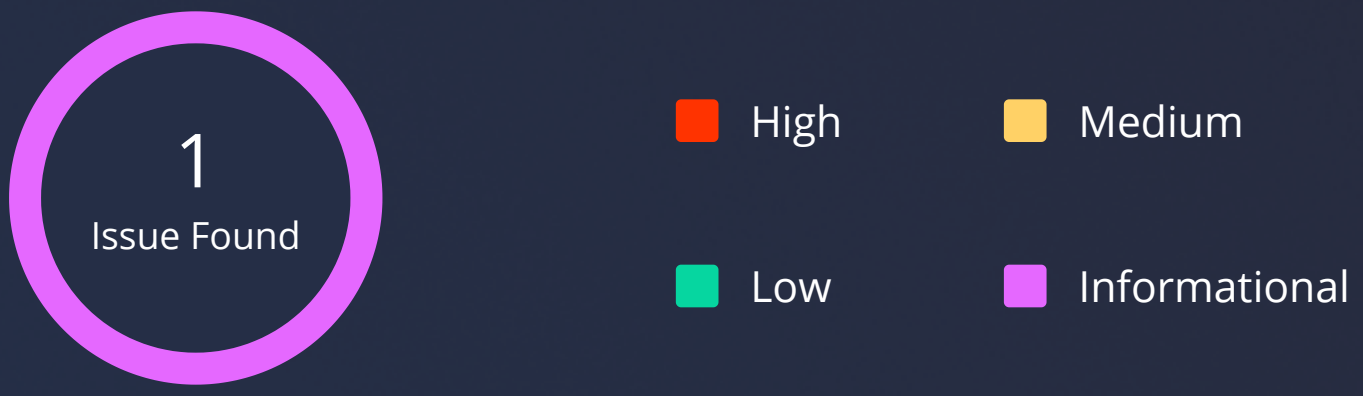
Closing Summary 10

Disclaimer 10

Executive Summary

Project Name	Shogun Multicall
Project URL	*https://shogun.fi/*
Overview	Inspired by the shōgun's unification of Japan, the Shogun Network will unveil a revolutionary approach to cross-chain liquidity, reshaping fragmented marketplaces in ways yet unseen.
Audit Scope	*https://github.com/shogun-network/MulticallRelayer-contract/tree/main/contracts*
Branch	Main
Contracts In-Scope	Contracts:- -ShogunMulticallRelayerV1.sol -ShogunMulticallV1.sol
Commit Hash	c46c234ed9f879fa28d5740132af846c95c962a8
Language	Solidity
Blockchain	Ethereum
Method	Manual Review, Automated Tools, Functional Testing
Review 1	24th June 2024 - 30th June 2024
Updated Code Received	NA
Review 2	NA
Fixed In	NA

Number of Security Issues per Severity



	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	1
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, Truffle, Solhint, Mythril, Slither, Solidity Static 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.

High Severity Issues

No issues were found.

Medium Severity Issues

No issues were found.

Low Severity Issues

No issues were found.

Informational Issues

1. Improper use may result in funds loss

Line **Function -** multicall

34 - 69

```
function multicall(
        Call[] memory calls,
        address swapTokenOut,
        address swapDestination,
        uint256 swapAmountOutMin
        ) external payable {
        uint256 initialAmount = _getBalance(swapTokenOut, swapDestination);

        for(uint256 i = 0; i < calls.length; i++) {
        Call memory call = calls[i];
+++

        (bool success, bytes memory data) = call.target.call{value: call.msgValue}
(call.data);
        if (!success) revert CallFailed(i, data);
        }
        }
```

Description

The contract is not supposed to have funds and only do the required aggregation and send the tokens to the target address. The contract is made flexible for aggregation in a way that complex calls can be constructed which helps to work gas cost efficiently with almost any protocol.

From a security standpoint, one of the most important parts is the function selector. The function parameters shouldn't be an issue since the contract is not supposed to hold funds and from the user's perspective the most important thing is at the end of execution the minimum amount of the requested asset has to be received. **With this, the issue was to deny transfers from approved funds if the initiator is not the user itself.**

However, it lacks adequate security checks to ensure that only the user who initiates the call can authorize transfers. This could lead to a situation where an attacker exploits the function to steal funds, particularly if the contract ends up holding funds due to an unexpected error or misuse.

Remediation

- **Add Authorization Checks:** Ensure that only the user who initiates the call can authorize transfers. Implement a mechanism to verify the caller's identity.
- **Restrict Fund Holding:** Ensure that the router does not hold any funds at any point. If funds are received, they should be immediately forwarded to the appropriate destination.

Status

Acknowledged

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 the Shogun Multicall. We performed our audit according to the procedure described above. Code Looks Good,only one Informational Severity Issue found,which the shogun team Acknowledged.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in Shogun Multicall smart contracts. 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 Shogun Multicall smart contracts. 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 the Shogun Multicall 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
July, 2024

For

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com