AudiTBlock

LIGHTSWAPInterfaceMulticall

v0.7.6+commit.7338295f

+ Low-Risk	+ Medium-Risk	+ High-Risk
low-risk code	medium-risk code	high-risk code

LIGHTSWAPInterfaceMulticall

Contract Deployed On nova.arbiscan.io

0x76a60fcd96cf0dd9e3642f921ac3d96a4dd6905b

Disclaimer

AudiTBlock is not responsible if a project turns out to be a scam, rug-pull or honeypot. We only provide a detailed analysis for your own research.

AudiTBlock is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your ownresearch.

The information provided in this audit is for informational purposes only and should not be considered investment advice. We does not endorse, recommend, support or suggest to invest in any project.

AudiTBlock can not be held responsible for when a project turns out to be a rug-pull, honeypot or scam.

Tokenomics

- Arbitrum Nova

Source Code

- AudiTBlock was complete audit phases to perform an audit based on the following smart contract:

- https://nova.arbiscan.io/address/0x76a60fcd96cf0dd9e3642f921ac3d96a4dd6905b#code

Snapshot 0.1

```
LIGHTSWAPInterfaceMulticall.multicall(LIGHTSWAPI
nterfaceMulticall.Call[]).target
(contracts/contract.sol#40) lacks a zero-check on :
        - (success,ret) = target.call{gas:
gasLimit}(callData) (contracts/contract.sol#46-48)
Reference:
https://github.com/crytic/slither/wiki/Detector-
Documentation#missing-zero-address-validation

LIGHTSWAPInterfaceMulticall.multicall(LIGHTSWAPI
nterfaceMulticall.Call[]) (contracts/contract.sol#34-52)
has external calls inside a loop: (success,ret) =
target.call{gas: gasLimit}(callData)
(contracts/contract.sol#46-48)
Reference:
https://github.com/crytic/slither/wiki/Detector-
Documentation/#calls-inside-a-loop

Low level call in
LIGHTSWAPInterfaceMulticall.multicall(LIGHTSWAPI
nterfaceMulticall.Call[]) (contracts/contract.sol#34-
52):
        - (success,ret) = target.call{gas:
gasLimit}(callData) (contracts/contract.sol#46-48)
Reference:
https://github.com/crytic/slither/wiki/Detector-
Documentation#low-level-calls
```

Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise,
after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed
condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5
Contracts/LIGHTSWAPInterfaceMulticall.sol	803cb7e955a88b85d1cba71fc81d2d62

Used Code from other Frameworks/Smart Contracts (direct imports)

Dependency / Import Path	Source Sha1 Hash
Contracts/interfaces	07c2c5e6c8692db10c7374847 0e3fd7815012246

0.2 SOLIDITY UNIT TESTING

```
Progress: 2 finished (of 2)
PASS ✓ ✓    Tested

✓ Check winning proposal
✓ Check winning proposal with return value


Result for tests Passed:
                                        0Time Taken: 0.24s
```

0.3 TESTING

```
 // SPDX-License-Identifier:MIT
pragma solidity =0.7.6;
pragma abicoder v2;

/// @notice A fork of Multicall2 specifically tailored for the Uniswap Interface
contract LIGHTSWAPInterfaceMulticall {
    struct Call {
        address target;
        uint256 gasLimit;
        bytes callData;
    }

    struct Result {
        bool success;
        uint256 gasUsed;
        bytes returnData;
    }

    function getCurrentBlockTimestamp() public view returns (uint256 timestamp) {
        timestamp = block.timestamp;
    }

    function getEthBalance(address addr) public view returns (uint256 balance) {
        balance = addr.balance;
    }

    function multicall(Call[] memory calls) public returns (uint256 blockNumber, Result[] memory returnData) {
        blockNumber = block.number;
        returnData = new Result[](calls.length);
        for (uint256 i = 0; i < calls.length; i++) {
            (address target, uint256 gasLimit, bytes memory callData) =
                (calls[i].target, calls[i].gasLimit, calls[i].callData);
            uint256 gasLeftBefore = gasleft();
            (bool success, bytes memory ret) = target.call{gas: gasLimit}(callData);
            uint256 gasUsed = gasLeftBefore - gasleft();
            returnData[i] = Result(success, gasUsed, ret);
        }
    }
}
```

- Compiler version =0.7.6 does not satisfy the r semver requirement
- Compiler version v2 does not satisfy the r semver requirement
- Avoid to make time-based decisions in your business logic
- Avoid to use low level calls.

0.1 Auto Debugging

```
block.chainid:
0xd05
block.coinbase:
0x0000000000000000000000000000000000000000
block.difficulty:
69762765929000
block.gaslimit:
708695
block.number:
1
block.timestamp:
1686904076
msg.sender:
0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
msg.sig:
0x60806040
msg.value:
0 Wei
tx.origin:
0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
block.basefee:
1 Wei (1)
```

Manual and Automated Vulnerability Test

CRITICAL ISSUES
During the audit, AudiTBlock experts found **0 medium Critical issues** in the code of the smart contract.

HIGH ISSUES
During the audit, AudiTBlock experts found **0 High issues** in the code of the smart contract.

MEDIUM ISSUES
During the audit, AudiTBlock experts found **0 Medium issues** in the code of the smart contract.

LOW ISSUES
During the audit, AudiTBlock experts found **0 Low issues** in the code of the smart contract.

INFORMATIONAL ISSUES
During the audit, AuditBlock experts
found **0 Informational issues** in the code of the smart contract.

SWC Attacks

ID	Title		Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✓
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✓
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✓
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✓
SWC-127	Arbitrary Jump with Function TypeVariable	CWE-695: Use of Low-Level Functionality	✓
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✓
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✓
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✓

ID	Title		Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✓
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✓
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✓
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✓
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✓
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✓
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✓
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✓

Owner privileges

- Verify Claims
- The contract block difficulty 69762765929000
  Status: tested and verified ✓
- Status: tested 1 and verified ✓
- Status: tested 2 and verified ✓
- Status: tested 3 and verified ✓
- Status: tested 4 and verified ✓
- Status: tested and verified ✓

Executive Summary

Two (2) independent AuditBlock experts performed an unbiased and isolated audit of the smart contract. The final debriefs

The overall code quality is good and not overloaded with unnecessary functions, these is greatly

benefiting the security of the contract. It correctly implemented widely used and reviewed contracts he main goal of the audit was to verify the claims regarding the security of the smart contract and the claims inside the scope of work.

During the audit, no issues were found after the manual and automated security testing.

Deployed On Nova Arbiscan

VERIFIED ✔

https://nova.arbiscan.io/address/0x76a60fcd96cf0dd9e3642f921ac3d96a4dd6905b#code