Smart Contract Audit Conclusion

by Vinh Le

22nd October 2022

This is the review of howeguo's Token-BulkSender repo. No critical bug found, but there exists several issue.

Table of Contents

Document properties

Version

Version	Date	Author	Description
0.1	Oct. 22, 2022	Vinh Le	Initial Draft

Summary

The following document provide the results of the audit performed by Vinh Le from Crystalize. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing general recommendations.

We have audited the Token-BulkSender, commit 8514f747cc876ce936d1a3de4b6e75dfaeb30f4b. The purpose of this DApp is to distribute tokens (i.e., ETH) to many receiver's wallet in one transactions, saving transactions fee.

Concretely, the following file was audited:

- ./BulkSender.sol

Findings Summary

Findings are categorized into different categories based on their degree of impact

- Critical, High Findings: No problem found
- Medium Findings:
    - Incorrect ERC20 interface
- Low Findings:
    - Missing events arithmetic
    - Missing events access control
    - Dead-code
    - Reentrancy vulnerabilities
- Informational/Gas Optimization Findings:
    - Incorrect versions of Solidity
    - State variables that could be declared constant
    - Public function that could be declared external

Medium Findings

Incorrect ERC20 interface

Proof of concept

Incorrect return values for ERC20 functions.

Impact

A contract compiled with Solidity > 0.4.22 interacting with these functions will fail to execute them, as the return value is missing.

- ERC20.transferFrom(address,address,uint256) (BulkSender.sol#57)
- ERC20.approve(address,uint256) (BulkSender.sol#58)
- ERC20Basic.transfer(address,uint256) (BulkSender.sol#51)
- BasicToken.transfer(address,uint256) (BulkSender.sol#72-76)
- StandardToken.transferFrom(address,address,uint256) (BulkSender.sol#91-96)
- StandardToken.approve(address,uint256) (BulkSender.sol#98-102)

Recommendation

Set the appropriate return values and types for the defined ERC20 function

Low Findings

Missing events arithmetic

Proof of concept

Detect missing events for critical arithmetic parameters.

Impact

BulkSender.setVIPFee(uint256) (BulkSender.sol#222-224) has no event, so it is difficult to track off-chain changes in the VIPFee.

Recommendation

Emit an event for critical parameter changes.

Missing events access control

Proof of concept

Detect missing events for critical access control parameters

Impact

Ownable.transferOwnership(address) (BulkSender.sol#125-129) has no event, so it is difficult to track off-chain owner changes.

Recommendation

Emit an event for critical parameter changes.

Dead-code

Proof of concept

Functions that are not used.

Impact

`dead_code` is not used in the contract, making the code's review more difficult.

- SafeMath.div(uint256,uint256) (BulkSender.sol#14-19)
- SafeMath.max256(uint256,uint256) (BulkSender.sol#35-37)
- SafeMath.max64(uint64,uint64) (BulkSender.sol#29-31)
- SafeMath.min256(uint256,uint256) (BulkSender.sol#38-40)

Recommendation

Remove unused functions

Reentrancy vulnerabilities

Proof of concept

Detection of the reentrancy bug. Only report reentrancies leading to out-of-order events.

Impact

Reentrancy in BulkSender.getBalance(address) (BulkSender.sol#153-163):

` LogGetToken` event emitted after external calls

balance = token.balanceOf(this) (BulkSender.sol#160)

token.transfer(_receiverAddress,balance) (BulkSender.sol#161)

Recommendation

Apply the [check-effects-interactions pattern]

Informational/Gas Optimization Findings

Incorrect versions of Solidity

Proof of concept

Pragma version^0.4.0 (BulkSender.sol#1) allows old versions, which is not recommended for deployment

Impact

`solc` frequently releases new compiler versions. Using an old version prevents access to new Solidity security checks.

Recommendation

Deploy with any of the following Solidity versions:

0.5.16 - 0.5.17, 0.6.11 - 0.6.12, 0.7.5 - 0.7.6, 0.8.16.

Consider using the latest version of Solidity for testing.

Public function that could be declared external

Proof of concept

`public` functions that are never called by the contract should be declared `external`, and its immutable parameters should be located in `calldata` to save gas

Impact

These functions should be declared external:

- BulkSender.addToVIPList(address[]) (BulkSender.sol#178-182)
- BulkSender.removeFromVIPList(address[]) (BulkSender.sol#187-191)
- BulkSender.sendEth(address[],uint256) (BulkSender.sol#322-324)
- BulkSender.bulksend(address[],uint256[]) (BulkSender.sol#329-331)
- BulkSender.bulkSendETHWithDifferentValue(address[],uint256[]) (BulkSender.sol#337- 339)
- BulkSender.bulkSendETHWithSameValue(address[],uint256) (BulkSender.sol#345-347)
- BulkSender.bulkSendCoinWithSameValue(address,address[],uint256) (BulkSender.sol#353-355)

- BulkSender.bulkSendCoinWithDifferentValue(address,address[],uint256[]) (BulkSender.sol#360-362)
- BulkSender.bulksendToken(address,address[],uint256[]) (BulkSender.sol#367-369)
- BulkSender.drop(address,address[],uint256) (BulkSender.sol#373-375)

Recommendation

Use the `external` attribute for functions never called from the contract and change the location of immutable parameters to `calldata` to save gas.

State variables that could be declared constant

Proof of concept

Constant state variables should be declared constant to save gas.

Impact

ERC20Basic.totalSupply (BulkSender.sol#49) should be constant

Recommendation

Add the `constant` attributes to state variables that never change

Conclusion

The use of the given smart contract is simple, with the size of the code being relatively small. Overall, the code demonstrates effective use of abstraction, separation of concern, and modularity. However, there are a few problems/vulnerabilities that need to be fixed at different security levels before any additional work or deployment.