Guild.xyz - Token BulkSender Review

By Anh Nguyen (Usua◎ Silver)

General Information

Resources:

The project repo was provided.

Project author:

Guild.xyz

Auditor:

Anh Nguyen (Usua◎ Silver)

Table of Contents

Summary

Token BulkSender

This DAPP was used to send token or ETH to many addresses in one transaction, and that can help user to save tx fee

Concretely, the following file was audited:

- ./BulkSender.sol

Scope

Code Repo

Commit

The commit reviewed was 8514f747cc876ce936d1a3de4b6e75dfaeb30f4b. The review covered the entire repository at this specific commit but focused on the BulkSender.sol file.

The review is a code review to identify potential vulnerabilities in the code. The reviewers did not investigate security practices or operational security and assumed that privileged accounts could be trusted. The reviewers did not evaluate the security of the code relative to a standard or specification. The review may not have identified all potential attack vectors or areas of vulnerability.

Crystalize makes no warranties regarding the security of the code and do not warrant that the code is free from defects. Crystalize does not represent nor imply to third party users that the code has been audited nor that the code is free from defects. By deploying or using the code, Guild.xyz and users agree to use the code at their own risk.

Code Evaluation Matrix

Category	Mark	Description
Access Control	Good	Access control applied where needed
Mathematics	Good	SafeMath library was used to check for overflow and underflow with simple math operations. No low-level bitwise operations are performed. There was no unusually complex math.
Compiler	Bad	Solidity compiler version ^0.4.0 was used. This version dated too far back. It lacks the additional security checks and improvement and may not be compatible with smart contracts that have solidity version > 0.4.22.
Libraries	Good	Only SafeMath and no external library is used. Fewer and simpler external dependencies is always a plus for security.
Documentation	Medium	Natspecs existed in many places and clarified part of what the code did. However, Many functions were not covered.
Monitoring	Medium	Functions that modified state variables are missing event emission.
Testing and verification	Bad	No testing provided.
Decentralization	Good	No external party access provided.

Findings Explanation

Findings are broken down into sections by their respective impact:

- Critical, High, Medium, Low impact
    - These are findings that range from attacks that may cause loss of funds, impact control/ownership of the contracts, or cause any unintended consequences/actions that are outside the scope of the requirements,
- Gas savings
    - Findings that can improve the gas efficiency of the contracts
- Informational
    - Findings including recommendations and best practices

No Critical or High Findings

Medium Findings

1. 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 functions.

Low Findings

1. 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. BulkSender.transferOwnership(address) should emit an event for: owner = newOwner (BulkSender.sol#127)

2. 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. BulkSender.setVIPFee(uint256) should emit an event for: VIPFee = _fee (BulkSender.sol#223)

3. 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)

```
153     function getBalance(address _tokenAddress) onlyOwner public {
154         address _receiverAddress = getReceiverAddress();
155         if (_tokenAddress == address(0)) {
156             require(_receiverAddress.send(address(this).balance));
157             return;
158         }
159         StandardToken token = StandardToken(_tokenAddress);
160         uint256 balance = token.balanceOf(this);
161         token.transfer(_receiverAddress, balance);
162         emit LogGetToken(_tokenAddress, _receiverAddress, balance)
163     }
```

The ` LogGetToken` events will be shown in an incorrect order, which might lead to issues for third parties.

Recommendation

Apply the check-effects-interactions pattern.

4. Dead-code

Proof of concept

Functions that are not used.

Impact

`dead_code` is not used in the contract, and make 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
- SafeMath.min64(uint64,uint64) (BulkSender.sol#32-34)

Recommendation

Remove unused functions.

Informational Findings

1. 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

The recommendations take into account:

- Risks related to recent releases
- Risks of complex code generation changes
- Risks of new language features
- Risks of known bugs

Use a simple pragma version that allows any of these versions. Consider using the latest version of Solidity for testing.

2. Reentrancy vulnerabilities

Proof of concept

Detection of the reentrancy bug. Only report reentrancy that is based on `transfer` or `send`.

Impact

`send` and `transfer` do not protect from reentrancies in case of gas price changes.

- `vipList[msg.sender] = true (BulkSender.sol#172)` State variables written after external calls: - require(bool)(_receiverAddress.send(msg.value)) (BulkSender.sol#171)

```
168     function registerVIP() payable public {
169         require(msg.value >= VIPFee);
170         address _receiverAddress = getReceiverAddress();
171         require(_receiverAddress.send(msg.value));
172         vipList[msg.sender] = true;
173     }
```

- `LogTokenBulkSent` emitted after external calls: - require(bool)(_to[i].send(_value)) (BulkSender.sol#248)

```
233     function ethSendSameValue(address[] _to, uint _value) internal {
234
235         uint sendAmount = _to.length.sub(1).mul(_value);
236         uint remainingValue = msg.value;
237
238         bool vip = isVIP(msg.sender);
239         if (vip) {
240             require(remainingValue >= sendAmount);
241         } else {
242             require(remainingValue >= sendAmount.add(txFee));
243         }
244         require(_to.length <= 255);
245
246         for (uint8 i = 1; i < _to.length; i++) {
247             remainingValue = remainingValue.sub(_value);
248             require(_to[i].send(_value));
249         }
250
251         emit LogTokenBulkSent(0x000000000000000000000000000000000000bEEF, msg.value);
252     }
```

- `LogTokenBulkSent` emitted after external calls: - require(bool)(_to[i].send(_value[i])) (BulkSender.sol#271)

```
254     function ethSendDifferentValue(address[] _to, uint[] _value) internal {
255
256         uint sendAmount = _value[0];
257         uint remainingValue = msg.value;
258
259         bool vip = isVIP(msg.sender);
260         if (vip) {
261             require(remainingValue >= sendAmount);
262         } else {
263             require(remainingValue >= sendAmount.add(txFee));
264         }
265
266         require(_to.length == _value.length);
267         require(_to.length <= 255);
268
269         for (uint8 i = 1; i < _to.length; i++) {
270             remainingValue = remainingValue.sub(_value[i]);
271             require(_to[i].send(_value[i]));
272         }
273         emit LogTokenBulkSent(0x000000000000000000000000000000000000bEEF, msg.value);
274
275     }
```

Recommendation

Apply the check-effects-interactions pattern.

Gas Optimization Findings

1. 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.

2. 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.

Final Remarks

With a focus on vulnerabilities with any external-call mechanics to the BulkSender.sol contract, I found nothing particularly worthy of note that was a critical exploit