Audit Report

Expart Bucks multisend

January 2022

Type	HRC20
Network	HARMONY
Address	0x2e6be092f02f9a9ca112c039b574f7f1e3495947
Audited by	© coinscope

Table of Contents

Contract Review

Contract Name	MultiSend
Compiler Version	0.7.6
Optimization	
Licence	
Explorer	https://explorer.harmony.one/address/0x2e6be092f02f9a9ca112c039b574f7f1e3495947
Source	contract.sol

Audit Updates

Initial Audit	27th January 2022
Corrected	

Contract Diagnostics

Critical Medium Minor

Severity	Code	Description
	CO	Code Optimization
	L01	Public Function could be Declared External
	L04	Conformance to Solidity Naming Conventions
	L09	Dead Code Elimination

CO - Code Optimization

Criticality	minor
Location	Contract.sol, lines are mentioned below

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

Line 860

There is an unused loop consuming unnecessary processing resources.

```
uint256 _value = msg.value;
for (uint8 i; i < _addresses.length; i++) {
    _value = _value.sub(_amounts[i]);

    // solhint-disable-next-line avoid-low-level-calls, avoid-call-value
    /*(success, ) = */_addresses[i].call{ value: _amounts[i] }("");
    // we do not care. caller should check sending results manually and re-send
if needed.
}
```

Line 855

- There should be a check that the size of the `_addresses` and `_amounts` is equal.
- There should be a check that guarantees that the user's funds are more than the accumulated amount of the `_amounts` array.

```
function multiTransfer_OST(address payable[] calldata _addresses, uint256[]
calldata _amounts)
```

Line 880

There are expressions that are not used, consuming unnecessary processing resources.

```
uint256 _value = msg.value;
_value = _value.sub(_amount1);
_value = _value.sub(_amount2);
```

Line 877

There should be a check that guarantees that the user's funds are more than the `amount1` and `amount2.`

```
function transfer2(address payable _address1, uint256 _amount1, address payable
_address2, uint256 _amount2)
  payable external whenNotPaused returns(bool)
```

Line 993

The `_amountSum` is not used, consuming unnecessary processing resources.

```
token.safeTransferFrom(msg.sender, address(this), _amountSum);
for (uint8 i; i < _addresses.length; i++) {
    _amountSum = _amountSum.sub(_amounts[i]);
    token.transfer(_addresses[i], _amounts[i]);
}
```

Line 983

- There should be a check that the size of the `_addresses` and `_amounts` is equal.
- There should be a check that guarantees that the user's funds are more than the accumulated amount of the `_amounts` array.

```
  function multiTransferToken_a4A(
    address _token,
    address[] calldata _addresses,
    uint256[] calldata _amounts,
    uint256 _amountSum
  ) payable external whenNotPaused
```

Line 1112

The `_amountSum` subtraction is not used, consuming unnecessary processing resources.

```
token.safeTransferFrom(msg.sender, address(this), _amountSum);
// bool success;
for (uint8 i; i < _addresses.length; i++) {
    _amountSum = _amountSum.sub(_amounts[i]);
    _value = _value.sub(_amountsEther[i]);
    token.transfer(_addresses[i], _amounts[i]);

    // solhint-disable-next-line avoid-low-level-calls, avoid-call-value
    /*(success, ) = */_addresses[i].call{ value: _amountsEther[i] }("");
    // we do not care. caller should check sending results manually and re-send
if needed.
}
```

Line 1099

- There should be a check that the size of the `_addresses` and `_amounts` is equal.
- There should be a check that the size of the `_amounts` and `_amountsEther` is equal.
- There should be a check that guarantees that the user's funds are more than the accumulated amount of the `_amounts` array.

```
  function multiTransferTokenEther(
    address _token,
    address payable[] calldata _addresses,
    uint256[] calldata _amounts,
    uint256 _amountSum,
    uint256[] calldata _amountsEther
  ) payable external whenNotPaused
```

Recommendation

Rewrite some code segments so the runtime will be more performant.

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L682,L673,L654 and 1 more

Description

Public functions that are never called by the contract should be declared external to save gas.

```
transferOwnership
renounceOwnership
owner
...
```

Recommendation

Use the external attribute for functions never called from the contract

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L1171,L1170,L1169 and 27 more

Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
_amountEther
_amount
_addresses
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L368,L352,L332 and 12 more

Description

Functions that are not used in the contract, and make the code's size bigger.

```
mod
div
add
...
```

Recommendation

Remove unused functions.

Contract Functions

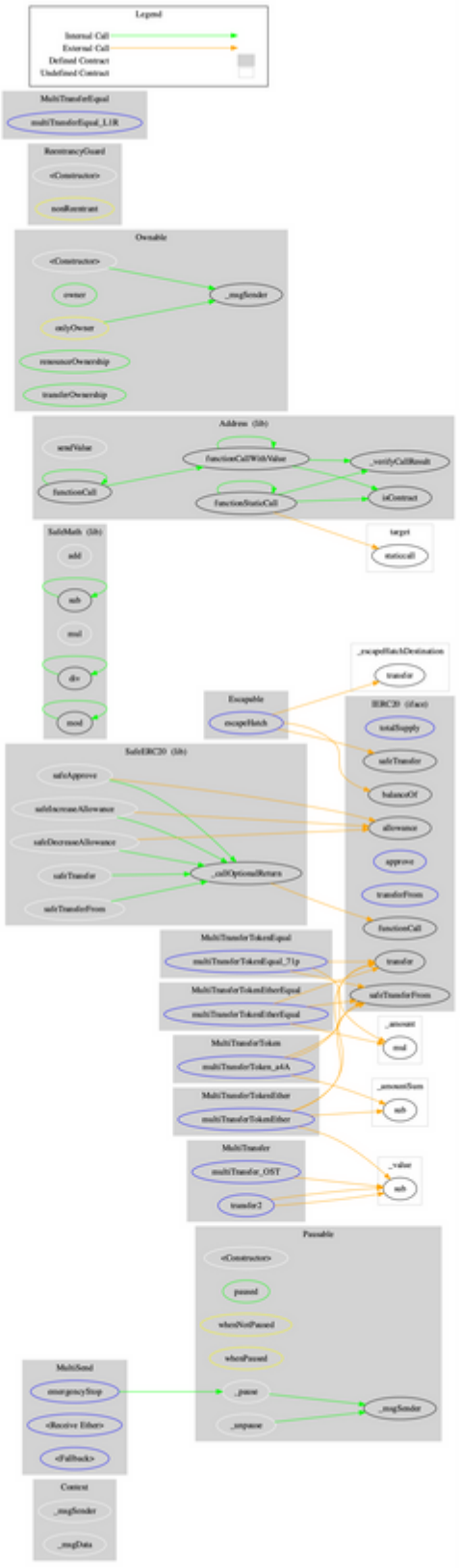
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Pausable	Implementation	Context		
	<Constructor>	Internal	✓	
	paused	Public		-
	_pause	Internal	✓	whenNotPaused
	_unpause	Internal	✓	whenPaused
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		

Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	_verifyCallResult	Private		
SafeERC20	Library			
	safeTransfer	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeApprove	Internal	✓	
	safeIncreaseAllowance	Internal	✓	
	safeDecreaseAllowance	Internal	✓	
	_callOptionalReturn	Private	✓	
Ownable	Implementation	Context		
	<Constructor>	Internal	✓	
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
ReentrancyGuard	Implementation			
	<Constructor>	Internal	✓	
Escapable	Implementation	Ownable, ReentrancyGuard		
	escapeHatch	External	✓	onlyOwner nonReentrant
MultiTransfer	Implementation	Pausable		
	multiTransfer_OST	External	Payable	whenNotPause

				d
	transfer2	External	Payable	whenNotPaused
MultiTransferEqual	Implementation	Pausable		
	multiTransferEqual_L1R	External	Payable	whenNotPaused
MultiTransferToken	Implementation	Pausable		
	multiTransferToken_a4A	External	Payable	whenNotPaused
MultiTransferTokenEqual	Implementation	Pausable		
	multiTransferTokenEqual_71p	External	Payable	whenNotPaused
MultiTransferTokenEther	Implementation	Pausable		
	multiTransferTokenEther	External	Payable	whenNotPaused
MultiTransferTokenEtherEqual	Implementation	Pausable		
	multiTransferTokenEtherEqual	External	Payable	whenNotPaused
MultiSend	Implementation	Pausable, Escapable, MultiTransfer, MultiTransferEqual, MultiTransferToken, MultiTransferTokenEqual, MultiTransferTokenEther, MultiTransfe		

		rTokenEther Equal		
	emergencyStop	External	✓	onlyOwner
	<Receive Ether>	External	Payable	-
	<Fallback>	External	Payable	-

Contract Flow



Summary

The contract contains functions that send funds to multiple addresses with many alternatives. Hence, it is a utility contract that does not contain functionality that may harm the users. The contract could embody some checks validating that the sender’s funds are equal to the accumulated amount that is issued. Some statements are not used by the contract logic. These statements could be removed. The amounts that have accumulated to contract from transactions that have failed can be moved to the user’s wallet using the escapeHatch function.

Disclaimer

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment.

Coinscope team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed.

The Coinscope team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Coinscope receive a payment to manipulate those results or change the awarding badge that we will be adding in our website.

Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token.

The Coinscope team disclaims any liability for the resulting losses.

About Coinscope

Coinscope is the leading early coin listing, voting and auditing authority firm. The audit process is analyzing and monitoring many aspects of the project. That way, it gives the community a good sense of security using an informative report and a generic score.

Coinscope is aiming to make crypto discoverable and efficient globally. It provides all the essential tools to assist users draw their own conclusions.

The Coinscope.co team

https://www.coinscope.co