BEOSIN

Blockchain Security

# Smart contract security audit report

**Audit Number：202107151418**

**Report Query Name: ramp-protocol**

**Smart Contract Link:**

https://github.com/RAMP-DEFI/ramp-protocol/tree/development

**Start Commit Hash：**

30edbfbbc3311131b676d0db532121089b74b734

**Finish Commit Hash：**

2687d2a8cd68da160d280fb155e497c1c97206e4

**Start Date：2021.07.01**

**Completion Date：2021.07.15**

**Overall Result：Pass**

**Audit Team: Beosin (Chengdu LianAn) Technology Co. Ltd.**

## Audit Categories and Results:

| No. | Categories | Subitems | Results |
|---|---|---|---|
| 1 | Coding Conventions | Compiler Version Security | Pass |
| | | Deprecated Items | Pass |
| | | Redundant Code | Pass |
| | | SafeMath Features | Pass |
| | | require/assert Usage | Pass |
| | | Gas Consumption | Pass |
| | | Visibility Specifiers | Pass |
| | | Fallback Usage | Pass |
| 2 | General Vulnerability | Integer Overflow/Underflow | Pass |
| | | Reentrancy | Pass |
| | | Pseudo-random Number Generator (PRNG) | Pass |
| | | Transaction-Ordering Dependence | Pass |
| | | DoS (Denial of Service) | Pass |
| | | Access Control of Owner | Pass |
| | | Low-level Function (call/delegatecall) Security | Pass |

| | | | |
|---|---|---|---|
| | | Returned Value Security | Pass |
| | | tx.origin Usage | Pass |
| | | Replay Attack | Pass |
| | | Overriding Variables | Pass |
| 3 | Business Security | Business Logics | Pass |
| | | Business Implementations | Pass |

Disclaimer: This report is made in response to the project code. No description, expression or wording in this report shall be construed as an endorsement, affirmation or confirmation of the project. This audit is only applied to the type of auditing specified in this report and the scope of given in the results table. Other unknown security vulnerabilities are beyond auditing responsibility. Beosin (Chengdu LianAn) Technology only issues this report based on the attacks or vulnerabilities that already existed or occurred before the issuance of this report. For the emergence of new attacks or vulnerabilities that exist or occur in the future, Beosin (Chengdu LianAn) Technology lacks the capability to judge its possible impact on the security status of smart contracts, thus taking no responsibility for them. The security audit analysis and other contents of this report are based solely on the documents and materials that the contract provider has provided to Beosin (Chengdu LianAn) Technology before the issuance of this report, and the contract provider warrants that there are no missing, tampered, deleted; if the documents and materials provided by the contract provider are missing, tampered, deleted, concealed or reflected in a situation that is inconsistent with the actual situation, or if the documents and materials provided are changed after the issuance of this report, Beosin (Chengdu LianAn) Technology assumes no responsibility for the resulting loss or adverse effects. The audit report issued by Beosin (Chengdu LianAn) Technology is based on the documents and materials provided by the contract provider, and relies on the technology currently possessed by Beosin (Chengdu LianAn). Due to the technical limitations of any organization, this report conducted by Beosin (Chengdu LianAn) still has the possibility that the entire risk cannot be completely detected. Beosin (Chengdu LianAn) disclaims any liability for the resulting losses.

The final interpretation of this statement belongs to Beosin (Chengdu LianAn).

## Audit Results Explained:

Beosin (Chengdu LianAn) Technology has used several methods including Formal Verification, Static Analysis, Typical Case Testing and Manual Review to audit three major aspects of smart contracts project ramp-protocol, including Coding Standards, Security, and Business Logic. **The ramp-protocol project passed all audit items. The overall result is Pass. The smart contracts is able to function properly.**

## Audit Contents:

### 1. Coding Conventions

Check the code style that does not conform to Solidity code style.

1.1 Compiler Version Security

- Description: Check whether the code implementation of current contract contains the exposed solidity compiler bug.
- Result: Pass

1.2 Deprecated Items

- Description: Check whether the current contract has the deprecated items.
- Result: Pass

1.3 Redundant Code

- Description: Check whether the contract code has redundant codes.
- Result: Pass

1.4 SafeMath Features

- Description: Check whether the SafeMath has been used. Or prevents the integer overflow/underflow in mathematical operation.
- Result: Pass

1.5 require/assert Usage

- Description: Check the use reasonability of 'require' and 'assert' in the contract.
- Result: Pass

1.6 Gas Consumption

- Description: Check whether the gas consumption exceeds the block gas limitation.
- Result: Pass

1.7 Visibility Specifiers

- Description: Check whether the visibility conforms to design requirement.
- Result: Pass

1.8 Fallback Usage

- Description: Check whether the Fallback function has been used correctly in the current contract.
- Result: Pass

### 2. General Vulnerability

Check whether the general vulnerabilities exist in the contract.

2.1 Integer Overflow/Underflow

- Description: Check whether there is an integer overflow/underflow in the contract and the calculation result is abnormal.
- Result: Pass

2.2 Reentrancy

- Description: An issue when code can call back into your contract and change state, such as withdrawing ETH.
- Result: Pass

2.3 Pseudo-random Number Generator (PRNG)

- Description: Whether the results of random numbers can be predicted.
- Result: Pass

2.4 Transaction-Ordering Dependence

- Description: Whether the final state of the contract depends on the order of the transactions.
- Result: Pass

2.5 DoS (Denial of Service)

- Description: Whether exist DoS attack in the contract which is vulnerable because of unexpected reason.
- Result: Pass

2.6 Access Control of Owner

- Description: Whether the owner has excessive permissions, such as malicious issue, modifying the balance of others.
- Result: Pass

2.7 Low-level Function (call/delegatecall) Security

- Description: Check whether the usage of low-level functions like call/delegatecall have vulnerabilities.
- Result: Pass

2.8 Returned Value Security

- Description: Check whether the function checks the return value and responds to it accordingly.
- Result: Pass

2.9 tx.origin Usage

- Description: Check the use secure risk of 'tx.origin' in the contract.
- Result: Pass

2.10 Replay Attack

- Description: Check whether the implement possibility of Replay Attack exists in the contract.
- Result: Pass

2.11 Overriding Variables

- Description: Check whether the variables have been overridden and lead to wrong code execution.
- Result: Pass

## 3. Business Security

Check whether the business is secure.

3.1 Business analysis of Contract RUSD and RToken.

(1) Basic Token Information of rUSD

| | |
|---|---|
| Token name | rUSD |
| Token symbol | rUSD |
| decimals | 18 |
| totalSupply | The initial supply is 0, mintable, burnable |
| Token type | ERC677 |

Table 1 Basic Token Information

(2) Introduction of rToken

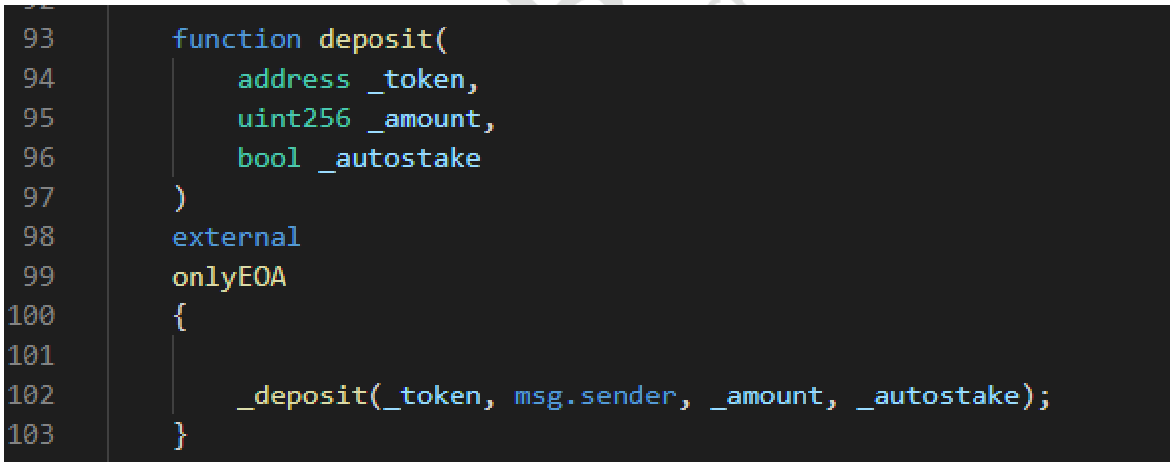
rToken is a token issued based on the deposit of tokens. When a user deposits tokens in the vault contract, the corresponding rToken will be mined and stake to the vault contract.

3.2 Contract of Vault

The Vault contract is responsible for storing the rToken of the user's stake and acts as a bridge between the bank contract and the strategies contracts.

(1) Deposit function

- Description: The *deposit* implements the function of the user's deposit of basic assets. The user deposits the basic assets in the strategy contract by calling the *deposit* function, and the strategy contract makes investment profits. The user will obtain the rToken corresponding to the token and force it to lock the position in the vault contract.



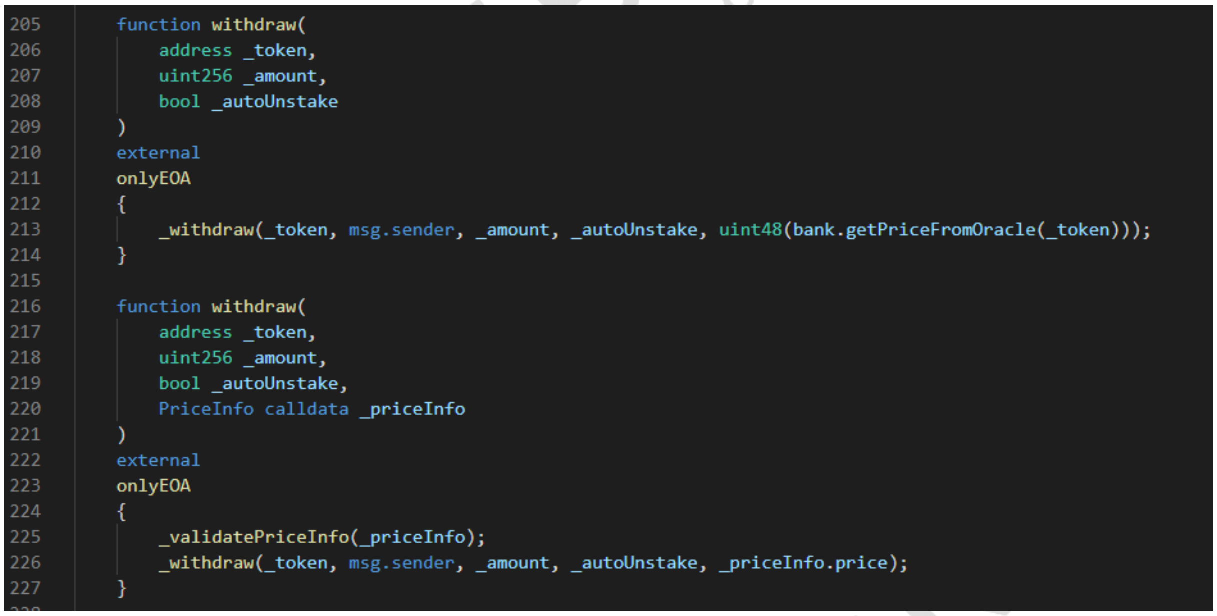
```
    function deposit(
        address _token,
        uint256 _amount,
        bool _autostake
    )
    external
    onlyEOA
    {

        _deposit(_token, msg.sender, _amount, _autostake);
    }
```

Figure 1 source code of *deposit*

- Related function: *deposit, _deposit*
- Result: Pass

(2) Withdraw function

- Description: The *withdraw* implements the function of user withdraw the deposited basic assets. The user destroys the rToken locked in the vault by calling the *withdraw* function and retrieves the underlying assets and the profit of the investment.



```
function withdraw(
    address _token,
    uint256 _amount,
    bool _autoUnstake
)
external
onlyEOA
{
    _withdraw(_token, msg.sender, _amount, _autoUnstake, uint48(bank.getPriceFromOracle(_token)));
}

function withdraw(
    address _token,
    uint256 _amount,
    bool _autoUnstake,
    PriceInfo calldata _priceInfo
)
external
onlyEOA
{
    _validatePriceInfo(_priceInfo);
    _withdraw(_token, msg.sender, _amount, _autoUnstake, _priceInfo.price);
}
```

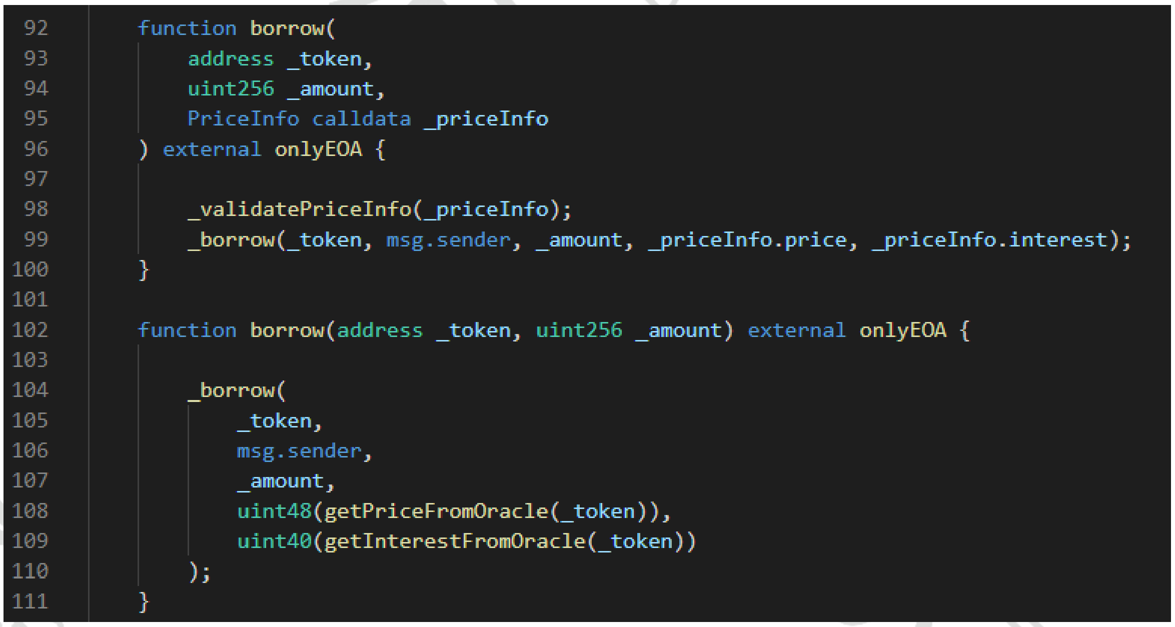
Figure 2 source code about *withdraw*

- Related function: *withdraw, _withdraw, _validatePriceInfo*
- Result: Pass

3.3 Contract of Bank

(1) Borrow function

- Description:The *borrow* implements the function of user borrowing. The user can call *borrow* to borrow rUSD, and the amount of borrowing depends on the value of the rToken locked by the user in the vault contract.



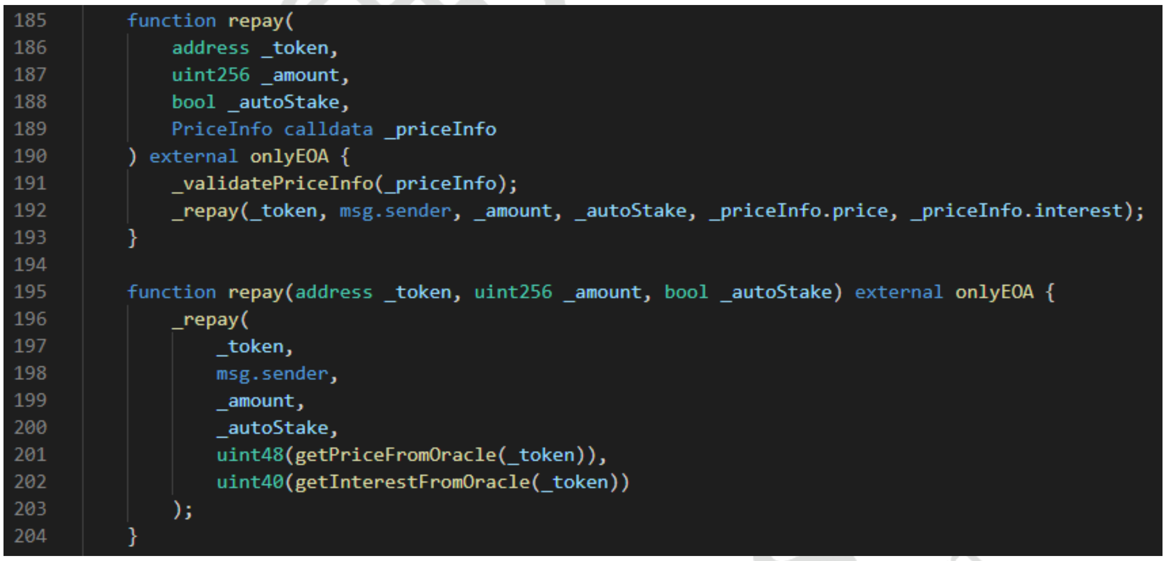
```
function borrow(
    address _token,
    uint256 _amount,
    PriceInfo calldata _priceInfo
) external onlyEOA {

    _validatePriceInfo(_priceInfo);
    _borrow(_token, msg.sender, _amount, _priceInfo.price, _priceInfo.interest);
}

function borrow(address _token, uint256 _amount) external onlyEOA {

    _borrow(
        _token,
        msg.sender,
        _amount,
        uint48(getPriceFromOracle(_token)),
        uint40(getInterestFromOracle(_token))
    );
}
```

Figure 3 source code of *borrow*

- Related function: *borrow, _validatePriceInfo, _borrow*
- Result: Pass

(2) Repay function

- Description: The *repay* implements the function of user to repay the loan. The user can repay the loaned rUSD and interest by calling the *repay* function. If the user does not repay the rUSD, the basic assets of the user deposit in the vault cannot be retrieved.



```
    function repay(
        address _token,
        uint256 _amount,
        bool _autoStake,
        PriceInfo calldata _priceInfo
    ) external onlyEOA {
        _validatePriceInfo(_priceInfo);
        _repay(_token, msg.sender, _amount, _autoStake, _priceInfo.price, _priceInfo.interest);
    }

    function repay(address _token, uint256 _amount, bool _autoStake) external onlyEOA {
        _repay(
            _token,
            msg.sender,
            _amount,
            _autoStake,
            uint48(getPriceFromOracle(_token)),
            uint40(getInterestFromOracle(_token))
        );
    }
```

Figure 4 source code of *repay*

- Related function: *repay, _validatePriceInfo, _repay*
- Result: Pass

## 4. Conclusion

Beosin(Chengdu LianAn) conducted a detailed audit on the design and code implementation of the smart contracts project ramp-protocol. The problems found by the audit team during the audit process have been notified to the project party and reached an agreement on the repair results, the overall audit result of the ramp-protocol project's smart contracts is **Pass**.

Beosin

BEOSIN

Blockchain Security

**Official Website**

https://lianantech.com

**E-mail**

vaas@lianantech.com

**Twitter**

https://twitter.com/Beosin_com