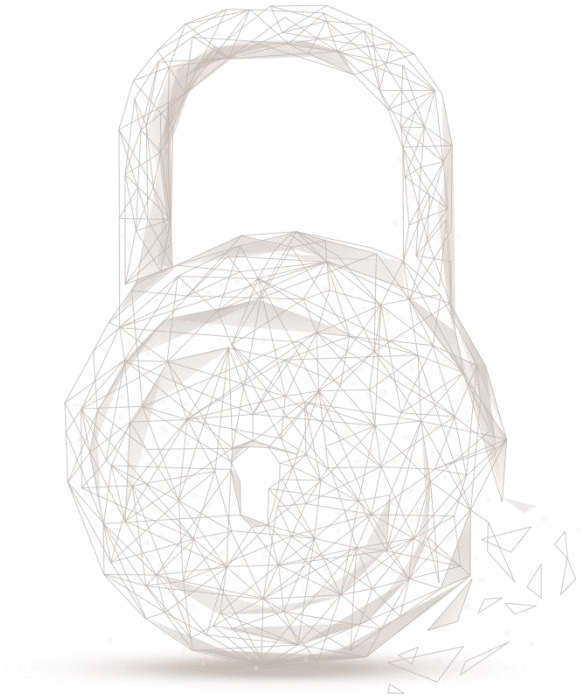
BEOSIN
Blockchain Security

Smart contract security audit report



Audit Number：202106111006

Report Query Name：AlpacaAccruingStrategy

Smart Contract Address Link:

https://github.com/RAMP-DEFI/ramp-protocol/blob/feature/hybrid_strategy/contracts/strategies/bsc/AlpacaAccruingStrategy.sol

Commit Hash:

Start: 76db18715bdceb37d2b8426a166834d5101ab3ff

Final: d074d2275446f6c65349347b3fec189fd86a4a06

Start Date：2021.05.28

Completion Date：2021.06.11

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

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 contract AlpacaAccruingStrategy, including Coding Standards, Security, and Business Logic. **The AlpacaAccruingStrategy contract passed all audit items. The overall result is Pass. The smart contract is able to function properly.**

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 BNB.
- 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

This contract is a strategic contract of vault and bank contracts, used to specify assets to invest in alpaca projects.

(1) onDepositHybrid

- Description: This function is used to trigger when the vault contract deposits the specified asset. The vault contract will inform the function of the amount to be invested. This function will send the funds to alpacaVault contract for investment, and reinvest the obtained ibToken.

```
102     function onDepositHybrid(
103         address _token,
104         address, // _account is not used in Accrueing strategies (no per-user ledger)
105         uint256 _amount
106     ) external override onlyVault returns (uint256) {
107         // Checks
108         require(_token == address(underlyingToken), "Wrong token!");
109 
110         underlyingToken.approve(address(alpacaVault), _amount);
111 
112         // Reinvest _amount in Alpaca Vault (returns ibToken)
113         alpacaVault.deposit(_amount);
114 
115         uint256 ibTokenReceived = ibToken.balanceOf(address(this));
116 
117         //update ibToken amount
118         totalIbToken = totalIbToken.add(ibTokenReceived);
119 
120         ibToken.approve(address(fairLaunch), ibTokenReceived);
121 
122         // We have now received ibToken now.
123         // We will stake it for getting more alpaca in fairLaunch contract.
124         // we dont know how many ibTokens we get. so take balanceOf ibTokeb and reinvest
125         fairLaunch.deposit(address(this), fairLaunchPoolId, ibTokenReceived);
126 
127         emit StrategyDeposit(_amount, ibTokenReceived);
128         return ibTokenReceived;
129     }
```

Figure 1 Source code of function *onDepositHybrid*

- Related functions: *approve*, *deposit*
- Result: Pass

(2) onWithdraw

- Description: This function is used to trigger when the Vault contract withdraws money. The vault contract specifies the recipient and the withdrawal amount. This contract will take out the corresponding amount of funds from the invested project and send it to the user.

```
142     function onWithdraw(
143         address _token,
144         address _account,
145         uint256 _amount // this is amount of underlying tokens
146     ) external override onlyVault {
147         // NOTE: We are NOT running "update" here, the Vault does this already.
148 
149         // 1) Determine amount of ibTokens for the _amount that we unstake and then burn for withdrawal
150         uint256 ibTokensToWithdraw = amountToShares(_amount);
151 
152         // 2) Unstake ibTokens from fairLaunch
153         // Remove _amount from alpacaVault.
154         // ibToken is being burned, and Alpaca is received.
155         // Since ibToken is reinvested in fairlaunch, first withdraw from fairlaunch
156         fairLaunch.withdraw(address(this), fairLaunchPoolId, ibTokensToWithdraw);
157 
158         // 3) Withdraw _amount from alpaca
159         // fairLaunch will give ibToken now, withdraw from alpacaVault
160         // This will burn ibToken and gives underlying Asset
161         alpacaVault.withdraw(ibTokensToWithdraw);
162 
163         // 4) Update ibToken total counter
164         totalIbToken = totalIbToken.sub(ibTokensToWithdraw);
165 
166         // 5) Apply fees
167         uint256 fee = _amount.mul(withdrawalFeePercentage).div(100);
168         if (fee > 0) underlyingToken.safeTransfer(devAddress, fee);
169 
170         // 6) Transfer to user
171         underlyingToken.safeTransfer(_account, _amount.sub(fee));
172 
173         // 7) Transfer all the Alpaca we received as rewards to the devAddress (not reinvesting it for now)
174         alpaca.safeTransfer(devAddress, alpaca.balanceOf(address(this)));
175 
176         // 8) Emit event
177         emit StrategyWithdraw(_account, _amount.sub(fee), fee);
178     }
```

Figure 2 Source code of function *onWithdraw*

- Related functions: *amountToShares*, *withdraw*, *safeTransfer*
- Result: Pass

(3) onLiquidate

- Description: This function is used to trigger when the bank contract liquidate, the vault contract will specify the amount of funds withdrawn, and the contract will withdraw the funds according to the specified amount and send them to the vault contract.

```
181     function onLiquidate(
182         address,
183         address,
184         uint256 _amount
185     ) external override onlyVault {
186 
187         // 1) Determine amount of ibTokens for the _amount that we unstake and then burn for withdrawal
188         uint256 ibTokensToWithdraw = amountToShares(_amount);
189 
190         // Since ibToken is reinvested in fairlaunch, first withdraw from fairlaunch
191         fairLaunch.withdraw(address(this), fairLaunchPoolId, ibTokensToWithdraw);
192 
193         // Remove _amount from alpacaVault.
194         // ibToken is being burned, and Alpaca is received.
195         alpacaVault.withdraw(ibTokensToWithdraw);
196 
197         //update ibToken amount
198         totalIbToken = totalIbToken.sub(ibTokensToWithdraw);
199 
200         uint underlyingTokenBalance = underlyingToken.balanceOf(address(this));
201 
202         // transfer alpaca to dev Address
203         alpaca.safeTransfer(devAddress, alpaca.balanceOf(address(this)));
204 
205         // send amount to vault
206         underlyingToken.safeTransfer(vault, underlyingTokenBalance);
207 
208         // Emit event for logging
209         emit Liquidated(_amount);
210 
211     }
```

Figure 3 Source code of function *onLiquidate*

- Related functions: *amountToShares, withdraw, safeTransfer*
- Result: Pass

(4) emergencyWithdraw

- Description: This function is used to trigger when the vault contract is withdrawn in an emergency. This function will withdraw all the investment and send it to the vault contract.

```
215     function emergencyWithdraw(address _token, bool) external override onlyVault returns (uint256 lastPoolAmount) {
216         // alpaca vault wont allow removal of tiny shares..(if totalSupply < 0.1)
217         // so  checking that in our contract
218         uint256 totalSupply = ibToken.totalSupply();
219         lastPoolAmount = totalStakePoolSize(); // saving the poolSize on emergency withdrawal
220         totalSupplyOnEmergencyWithdrawal = totalSupply; // saving the totalSupply of ibToken on emergency withdrawal
221
222         // first remove ibToken from fairLaunch
223         fairLaunch.withdraw(address(this), fairLaunchPoolId, totalIbToken);
224         totalIbToken = 0; //mark totalIbToken as zero (we removed everything from strategy)
225
226         // remove all stake from alpaca Strategy
227         alpacaVault.withdraw(ibToken.balanceOf(address(this)));
228
229         uint256 withdrawAmount = underlyingToken.balanceOf(address(this));
230
231         // transfer funds back to vault
232         underlyingToken.safeTransfer(
233             address(vault), //vault
234             underlyingToken.balanceOf(address(this))
235         );
236
237         uint256 alpacaAccReward = alpaca.balanceOf(address(this));
238
239         // transfer alpaca to dev address (alpaca belongs to ramp. so transfering to dev address instead vault)
240         alpaca.safeTransfer(
241             address(devAddress), // alpaca is for ramp. So transfer it to devAddress
242             alpacaAccReward
243         );
244
245         emit EmergencyWithdraw(_token, withdrawAmount, alpacaAccReward);
246     }
```

Figure 4 Source code of *emergencyWithdraw*

- Related functions:*withdraw*, *totalStakePoolSize*
- Result: Pass

4. Conclusion

Beosin(Chengdu LianAn) conducted a detailed audit on the design and code implementation of the smart contract AlpacaAccruingStrategy. The contract AlpacaAccruingStrategy passed all audit items, The overall audit result is **Pass.**

Official Website

https://lianantech.com

E-mail

vaas@lianantech.com

Twitter

https://twitter.com/Beosin_com