BEOSIN

Blockchain Security

Smart contract security audit report

Audit Number：202104121525

Report Query Name：coinwind-vault

Audit Project Name：coinwind-vault

Audit Project Contract Info：

Audit project file hash (SHA256)	241e3f4c28c37eed35a667191d1665188934c8740f47e0763c3789766e1ba8b9

Start Date：2021.03.05

Completion Date：2021.03.17

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

Note: Audit results and suggestions in code comments

Disclaimer: 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 coinwind-vault, including Coding Standards, Security, and Business Logic. **The coinwind-vault project passed all audit items. The overall result is Pass.** The smart contract 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.

  The compiler version specified in the smart contract of this project is 0.6.12, and the contract is compiled with this version of the compiler without any compiler warning.
- Safety Suggestion: None
- Fix Result: Ignored
- Result: Pass

1.2 Deprecated Items

- Description: Check whether the current contract has the deprecated items.
- Safety Suggestion: None
- Result: Pass

1.3 Redundant Code

- Description: Check whether the contract code has redundant codes.
- Safety Suggestion: None
- Result: Pass

1.4 SafeMath Features

- Description: Check whether the SafeMath has been used. Or prevents the integer overflow/underflow in mathematical operation.
- Safety Suggestion: None
- Result: Pass

1.5 require/assert Usage

- Description: Check the use reasonability of 'require' and 'assert' in the contract.
- Safety Suggestion: None
- Result: Pass

1.6 Gas Consumption

- Description: Check whether the gas consumption exceeds the block gas limitation.
- Safety Suggestion: None
- Result: Pass

1.7 Visibility Specifiers

- Description: Check whether the visibility conforms to design requirement.
- Safety Suggestion: None
- Result: Pass

1.8 Fallback Usage

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

- Safety Suggestion: None
- 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.
- Safety Suggestion: None
- Result: Pass

2.2 Reentrancy

- Description: An issue when code can call back into your contract and change state, such as withdrawing HT.
- Safety Suggestion: None
- Result: Pass

2.3 Pseudo-random Number Generator (PRNG)

- Description: Whether the results of random numbers can be predicted.
- Safety Suggestion: None
- Result: Pass

2.4 Transaction-Ordering Dependence

- Description: Whether the final state of the contract depends on the order of the transactions.
- Safety Suggestion: None
- Result: Pass

2.5 DoS (Denial of Service)

- Description: Whether exist DoS attack in the contract which is vulnerable because of unexpected reason.
- Safety Suggestion: None
- Result: Pass

2.6 Access Control of Owner

- Description: Whether the owner has excessive permissions, such as malicious issue, modifying the balance of others.
- Safety Suggestion: None
- Result: Pass

2.7 Low-level Function (call/delegatecall) Security

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

- Safety Suggestion: None
- Result: Pass

2.8 Returned Value Security

- Description: Check whether the function checks the return value and responds to it accordingly.
- Safety Suggestion: None
- Result: Pass

2.9 tx.origin Usage

- Description: Check the use secure risk of 'tx.origin' in the contract.
- Safety Suggestion: None
- Result: Pass

2.10 Replay Attack

- Description: Check the weather the implement possibility of Replay Attack exists in the contract.
- Safety Suggestion: None
- Result: Pass

2.11 Overriding Variables

- Description: Check whether the variables have been overridden and lead to wrong code execution.
- Safety Suggestion: None
- Result: Pass

3. Business Audit

3.1 The ControllerHub Contract Audit

3.1.1 Contract owner permission management

- Description: The highest permission owner of this contract (the contract deployer by default) can call the *transferOwnership* function to transfer the owner permission to the specified non-zero address; or call the *renounceOwnership* function to renounce the owner permission; call the *setGovernance* function to set the Governance contract address.
- Related functions: *transferOwnership*, *renounceOwnership*, *setGovernance*
- Result: Pass

3.1.2 Governance contract system governance

- Description: The Governance of this contract can call the relevant functions to set the key system parameters of the contract and contract address, such as: call the *setGovernance* function to update the Governance address; call the *setRewardAccount* function to set the contract's income address; call The *setPause* function changes the pause state of the contract (after the contract is suspended, earn will not be possible); call the *inCaseTokensGetStuck* function to withdraw the HRC-20 token at the specified

contract address; call the *setMdexTokenAddr* function to set the MDX token address; call the *setVault* function to set the vault address.

- Related functions: *setGovernance*, *setRewardAccount*, *setPause*, *inCaseTokensGetStuck*, *setMdexTokenAddr*, *setVault*
- Result: Pass

3.1.3 Deposit strategy management

- Description: The contract's Governance address can call the contract's *addStrategy* and *removeStrategy* functions to add and remove deposit strategies; it can also call the *setStrategyList* function to directly update the strategy list.

As shown in Figure 1, the caller of the *removeStrategy* function must be the Governance contract; but the corresponding Governance contract has the corresponding function annotated(as shown in Figure 2), then this function will not be called (Governance contract can modify the Governance of this contract to the account address, and then the account address can call the *removeStrategy* function to remove s*trategy*).

```
119     function removeStrategy(address _strategy) external onlyGovernance{
120         require(_strategy != address(0), "got zero address");
121         //提现金额到vault
122         IStrategy(_strategy).withdrawAll();
123         uint b = IStrategy(_strategy).balanceOf();
124         require(b == 0, "withdraw all lp not clear");
125 
126         for(uint i = 0; i < strategieList.length; i++) {
127             if(strategieList[i] != address(0) && _strategy == strategieList[i]) {
128                 delete strategieList[i];
129                 break;
130             }
131         }
132     }
```

Figure 1 The source code of removeStrategy function

```
312     // function removeStrategy(address targetAddress, address _strategy) external onlyGovernance {
313     //     IControllerHub(targetAddress).removeStrategy(_strategy);
314     // }
```

Figure 2 Annotated removeStrategy function

- Related functions: *addStrategy*, *removeStrategy*, *setStrategyList*
- Safety Suggestion: None
- Result: Pass

3.1.4 Deposit tokens

- Description: As shown in the figure below, any user can call the *earn* function of this contract to deposit the specified token. This function will traverse the list of strategies, select the strategy that supports the specified token *_token*, and deposit tokens in the specified strategy contract according to the token balance of the vault contract.

```
244     function earn(address _token) public {
245         if(paused){···
247         }
248
249         IStrategy _strategy;
250         // 策略对应的币对
251         address token1;
252         address token2;
253         // 对应币对的erc20余额
254         uint balance1 = IHubPool(vault).available(_token);
255         if(balance1 <= 0) {···
257         }
258         uint balance2;
259         // 尝试所有策略
260         for(uint i = 0; i < strategieList.length; i++) {
261             if(strategieList[i] == address(0)) {
262                 continue;
263             }
264
265             _strategy = IStrategy(strategieList[i]);
266             if(!_strategy.contain(_token)) {···
268             }
269
270             // 策略停止使用
271             if(_strategy.paused()) {···
273             }
274
275             // 策略希望提供的币种
276             (token1, token2) =  _strategy.want();
277             // 两个币种在金库中的余额
278             balance1 = IHubPool(vault).available(token1);
279             balance2 = IHubPool(vault).available(token2);
280             if(balance1 <= 0 || balance2 <= 0) {
281                 continue;
282             }
283
284             // 有资产将所有资产给到策略
285             IERC20(token1).safeTransferFrom(vault, strategieList[i], balance1);
286             IERC20(token2).safeTransferFrom(vault, strategieList[i], balance2);
287
288             _strategy.deposit();
289         }
290     }
```

Figure 3 The source code of *earn* function

In addition, the Governance contract can also call the *govEarn* function of this contract to make a specified amount of deposit in the strategy.

```
230     function govEarn(uint256 _sid, uint256 amount1, uint256 amount2) public onlyGovernance{
231         require(_sid < strategieList.length, "strategy error");
232         require(strategieList[_sid] != address(0), "strategy not exists");
233
234         IStrategy _strategy = IStrategy(strategieList[_sid]);
235         // 策略希望提供的币种
236         (address token1, address token2) =  _strategy.want();
237         //直接划钱到策略，记得先调vault授权
238         IERC20(token1).safeTransferFrom(vault, strategieList[_sid], amount1);
239         IERC20(token2).safeTransferFrom(vault, strategieList[_sid], amount2);
240         _strategy.deposit();
241     }
```

Figure 4 The source code of *govEarn* function

- Related functions: *earn*, *govEarn*
- Safety Suggestion: None
- Result: Pass

3.1.5 Release deposit tokens

- Description: The Governance of the contract can call related functions of the contract to release the deposit tokens.

The *withdrawAll* function is used to release all the deposit tokens in the current list that support the specified token strategy in full.

```
167     function withdrawAll(address _token) public onlyGovernance{
168         address _strategy;
169         for(uint i=0; i<strategieList.length; i++){
170             _strategy = strategieList[i];
171             //判断策略币对是否有此币种
172             if(_strategy != address(0) && IStrategy(_strategy).contain(_token)){
173                 IStrategy(_strategy).withdrawAll();
174             }
175         }
176     }
```

Figure 5 The source code of withdrawAll function

The *withdrawLp* function is used to release a specified address and a specified number of tokens. The function will traverse all strategies from back to front and select a strategy that supports the specified token to release until the cumulative amount released reaches *_amount*.

```
179     function withdrawLp(address _token, uint _amount) public {
180         require(msg.sender == vault || msg.sender == governance, "!vault");
181         require(address(0) != vault, "!vault");
182         require(_amount > 0, "amount error");
183         require(strategieList.length > 0, "strategie is empty");
184 
185         IStrategy _strategy;
186         //策略实际转到vault的数量
187         uint r;
188 
189         //从收益低的策略开始释放
190         uint i = strategieList.length;
191         while(i >= 0){
192             if(i == 0){
193                 break;
194             }else{
195                 i = i - 1;
196             }
197 
198             if(strategieList[i] == address(0)) {
199                 continue;
200             }
201 
202             _strategy = IStrategy(strategieList[i]);
203             //判断策略币对是否有此币种
204             if(!_strategy.contain(_token)) {
205                 continue;
206             }
207 
208             r = _strategy.withdraw(_token, _amount);
209             if(r >= _amount){
210                 break;
211             }
212             _amount = _amount.sub(r);
213         }
214     }
```

Figure 6 The source code of withdrawLp function

The *govWithdraw* function is used to release a specified strategy, a specified address, and a specified number of deposit tokens.

```
217     function govWithdraw(uint256 _sid, address token, uint256 amount) public onlyGovernance{
218         require(_sid < strategieList.length, "strategy error");
219         require(strategieList[_sid] != address(0), "strategy not exists");
220         require(token != address(0), "token is zero");
221 
222         IStrategy _strategy = IStrategy(strategieList[_sid]);
223         // 策略希望提供的币种
224         if(_strategy.contain(token)){
225             _strategy.withdraw(token, amount);
226         }
227     }
```

Figure 7 The source code of *govWithdraw* function

- Related functions: *withdrawAll, withdrawLp, govWithdraw*
- Safety Suggestion: None
- Result: Pass

3.1.6 Withdraw deposit rewards

- Description: Vault contract can call the *withdrawPending* function of this contract to withdraw deposit rewards. This function will only be called when the vault contract is depositing or withdrawing assets; and the debt will be updated in time after the call.

  As shown in the figure below, the function will call the *withdrawMDXReward* function on the strategy contract that supports the specified token to receive the deposit reward (MDX token), and then distribute this part of the token to the user and the platform income address.

```
332     function withdrawPending(address token, address user, uint256 userPending, uint256 govPending) public returns (bool){
333         require(msg.sender == vault , "!vault");
334         require(address(0) != rewardAccount, "rewardAccount address is zero");
335
336         uint256 total = userPending.add(govPending);
337         uint256 balance = IERC20(mdxToken).balanceOf(address(this));
338
339         if(balance < total){
340             IStrategy _strategy;
341             //把mdx的收益释放
342             for(uint i = 0; i < strategieList.length; i++) {
343                 if(strategieList[i] == address(0)) continue;
344                 _strategy = IStrategy(strategieList[i]);
345                 if(!_strategy.contain(token)) {
346                     continue;
347                 }
348                 _strategy.withdrawMDXReward();
349             }
350         }
351
352         balance = IERC20(mdxToken).balanceOf(address(this));
353         require(balance >= total, "profit token Insufficient");
354
355         if(userPending > 0) {
356             require(address(0) != user, "user address is zero");
357             // 给用户发
358             IERC20(mdxToken).safeTransfer(user, userPending);
359         }
360
361         if(govPending > 0) {
362             // 给平台分润
363             IERC20(mdxToken).safeTransfer(rewardAccount, govPending);
364         }
365         return true;
366     }
```

Figure 8 The source code of *withdrawMDXReward* function

- Related functions: *withdrawPending, withdrawMDXReward*
- Safety Suggestion: None
- Result: Pass

3.2 The GovernanceHub Contract Audit

3.2.1 Contract owner permission management

- Description: The highest permission owner of this contract (the contract deployer by default) can call the *transferOwnership* function to transfer the owner permission to the specified non-zero address; or call the *renounceOwnership* function to renounce the owner permission.
- Related functions: *transferOwnership*, *renounceOwnership*,
- Result: Pass

3.2.2 Governance permission management

- Description: The contract administrator owner can call the *addGovernance* function to add the specified address as the government; call the *removeGovernance* function to remove the specified address from the government list; or call the *resetGovernance* function to reset the government list.

In addition, the governance of this contract can operate all the business of this project. It is recommended that the address be set as the address of the governance contract after the project goes online, and it cannot be changed at will.

- Related functions: *addGovernance*, *removeGovernance*, *resetGovernance*
- Safety Suggestion: None
- Result: Pass

3.2.3 Administrator withdrawal

- Description: The contract's administrator owner can call the *adminWithdraw* function to withdraw tokens. As shown in the figure below, if the target address is not this contract, the *liquidityWithdraw* function on the target contract will be called (the target contract is vault); if it is this contract, the token will be sent directly to the caller.

```
186     function adminWithdraw(address targetAddress, address token, uint _amount) public onlyOwner{
187         require(targetAddress != address(0), "targetAddress is the zero address");
188         if(address(this) != targetAddress){
189             ISuperOwner(targetAddress).liquidityWithdraw(token, _amount);
190         }
191         IERC20(token).safeTransfer(msg.sender, _amount);
192     }
```

Figure 9 The source code of *adminWithdraw* function

The administrator of the contract can call the *inCaseTokensGetStuck* function to withdraw tokens. As shown in the figure below, if the target address is not this contract, the *inCaseTokensGetStuck* function on the target contract will be called (the target contract can be Strategy, vault, controller); if it is this contract, the tokens will be sent directly to the caller.

```
195     function inCaseTokensGetStuck(address targetAddress, address token, uint _amount) public onlyOwner{
196         require(targetAddress != address(0), "targetAddress is the zero address");
197         if(address(this) != targetAddress){
198             ISuperOwner(targetAddress).inCaseTokensGetStuck(msg.sender, token, _amount);
199         }
200         else{
201             IERC20(token).safeTransfer(msg.sender, _amount);
202         }
203     }
```

Figure 10 The source code of *inCaseTokensGetStuck* function

- Related functions: *adminWithdraw*, *inCaseTokensGetStuck*
- Safety Suggestion: None
- Result: Pass

3.2.4 Contract governance

- Description: At present, there are two kinds of management permission in this contract: the highest permission owner and the operation permission governance. Among them, this contract implements multiple functions that can only be called by the owner to manage key system parameters (such as vault address, income address, etc.) in the OwnerHub sub-contract; implements multiple functions that can only be called by the governance in other sub-contracts to manage the business functions of the system.

Note: The *removeStrategy* function is commented out in the GovernmentHubController contract, which results in the system's strategy cannot be removed.

```
311     // 删除策略
312     // function removeStrategy(address targetAddress, address _strategy) external onlyGovernance {
313     //     IControllerHub(targetAddress).removeStrategy(_strategy);
314     // }
```

Figure 11 Annotated removeStrategy function

- Safety Suggestion: None
- Result: Pass

3.3 The HubPool Contract Audit

3.3.1 Contract owner permission management

- Description: The highest permission owner of this contract (the contract deployer by default) can call the *transferOwnership* function to transfer the owner permission to the specified non-zero address; or call the *renounceOwnership* function to renounce the owner permission; call the *setGovernance* function to set the Governance contract address.
- Related functions: *transferOwnership*, *renounceOwnership*, *setGovernance*
- Safety Suggestion: None
- Result: Pass

3.3.2 Set key system parameters

- Description: As the governance contract of this project, Governance contract can call related functions of this contract to set contract key system parameters.
- Related functions: setLiquidityAddress, setPause, setController, setMin, setEarnLowerlimit, setTotalAmountLimit, setProfit, setSwapMiningAddr
- Safety Suggestion: None
- Result: Pass

3.3.3 Add token pledge pool

- Description: The governance contract Governance can call the *add* function of this contract to add a pledge pool. Although the developers of this function have noticed that tokens cannot be added repeatedly, it is still recommended to limit the code to avoid mistaken operations and repeated addition of tokens, which will affect the income of pledged users.
- Related functions: *add*
- Safety Suggestion: None
- Result: Pass

3.3.4 Update pledge pool data

- Description: When user deposit or withdraw token to the pledge pool by calling corresponding function , it will call the contract's *updatePool* function to update the data of the pledge pool. As shown

in the figure below, the function calls the *getMdxBlockReward* function to receive the MDX cumulative income of the specified pledge pool; and then calculates the users and platforms under a single token based on the total amount of tokens in the corresponding pledge pool, the distribution proportion and the last update data, accumulative income; and update the relevant data of the pledge pool to the latest.

```
274    function updatePool(uint256 _pid) override public {
275        PoolInfo storage pool = poolInfo[_pid];
276        if (block.number <= pool.lastRewardBlock) {
277            return;
278        }
279
280        // token目前累计的总mdx数量
281        uint256 blockReward = getMdxBlockReward(address(pool.token));
282        if (blockReward <= 0) {
283            return;
284        }
285
286        if(pool.accMdxShare >= blockReward) {
287            // 没有生成mdx奖励
288            pool.lastRewardBlock = block.number;
289            return;
290        }
291
292        if (pool.totalAmount == 0) {
293            pool.accMdxShare = blockReward;
294            pool.lastRewardBlock = block.number;
295            return;
296        }
297
298        // 单个token获取的mdx数量 =  (blockReward-pool.accMdxShare)/2*(pool.profit/max)/pool.totalAmount
299        // 计算用户收益率 除以2 再除 质押总量 再除万分比
300        uint256 divisor = pool.totalAmount.mul(20000);
301
302        //平台总收益 增量
303        govTotalProfit = (blockReward.sub(pool.accMdxShare)).mul(max.sub(pool.profit)).div(20000).add(govTotalProfit);
304        //用户总收益 增量
305        userTotalProfit = (blockReward.sub(pool.accMdxShare)).mul(pool.profit).div(20000).add(userTotalProfit);
306
307        // 计算每个token获取的mdx数量，放大12倍精度来计算
308        uint256 mdxReward = (blockReward.sub(pool.accMdxShare)).mul(1e12).div(divisor);
309        // 单个token累计的用户收益
310        pool.accMdxPerShare = pool.accMdxPerShare.add(mdxReward.mul(pool.profit));
311        // 单个token累计的平台分润
312        pool.govAccMdxPerShare = pool.govAccMdxPerShare.add( mdxReward.mul(max.sub(pool.profit)));
313
314        // 最后一块收益率
315        pool.lastRewardBlockProfit = mdxReward.mul(pool.profit).mul(max).div(block.number.sub(pool.lastRewardBlock));
316        // 最新结算收益的块高
317        pool.lastRewardBlock = block.number;
318        // 从第0块到lastRewardBlock 累计产生的mdx收益
319        pool.accMdxShare = blockReward;
320
321    }
```

Figure 12 The source code of updatePool function

- Related functions: *updatePool*
- Safety Suggestion: None
- Result: Pass

3.3.5 Token pledge

- Description: The user can call the *deposit* function to pledge a specified number of tokens; also call the *depositAll* function to pledge the entire balance of tokens; call the *depositWithPid* function to pledge the pledge pool with the specified id.

The first two of the above functions will eventually call the *depositWithPid* function. The source code is as shown in the figure below. After checking the basic parameters, the function will call the *updatePool* function to update the corresponding pledge pool data; if the corresponding user has already pledged, it will first settle the pledge rewards; if the number of tokens pledged this time is not 0, the transfer operation and pledge data operation update will be performed (the pledge quantity is 0, which means that only the reward is received); then, the *earn* function is called to earn the excess tokens; finally update the debt information of users and platforms.

```
357     function depositWithPid(uint256 _pid, uint256 _amount) public notPause {
358         require(_amount >= 0, "deposit: not good");
359         PoolInfo storage pool = poolInfo[_pid];
360         UserInfo storage user = userInfo[_pid][msg.sender];
361         if(pool.totalAmountLimit > 0){
362             //限制投资总量
363             require(pool.totalAmountLimit >= (pool.totalAmount + _amount), "deposit amount limit");
364         }
365         updatePool(_pid);
366 
367         // 先结算用户先前的投资
368         if (user.amount > 0) {
369             // 给用户发放收益与平台分润
370             uint256 pendingAmount = user.amount.mul(pool.accMdxPerShare).div(1e12).sub(user.rewardDebt);
371             //uint256 pendingGovAmount = user.amount.mul(pool.govAccMdxPerShare).div(1e12).sub(user.govRewardDebt);
372             uint256 pendingGovAmount = govTotalProfit.sub(govTotalSendProfit);
373             safeMdxTransfer(address(pool.token), msg.sender, pendingAmount, pendingGovAmount);
374         }
375 
376         // 执行扣用户的token
377         if (_amount > 0) {
378             uint256 beforeToken = pool.token.balanceOf(address(this));
379             pool.token.safeTransferFrom(msg.sender, address(this), _amount);
380             uint256 afterToken = pool.token.balanceOf(address(this));
381             _amount = afterToken.sub(beforeToken);
382 
383             if(_amount > 0) {
384                 user.amount = user.amount.add(_amount);
385                 pool.totalAmount = pool.totalAmount.add(_amount);
386             }
387         }
388 
389         // 重新触发投资
390         earn(address(pool.token));
391 
392         // 更新用户负债
393         user.rewardDebt = user.amount.mul(pool.accMdxPerShare).div(1e12);
394         user.govRewardDebt = user.amount.mul(pool.govAccMdxPerShare).div(1e12);
395         emit Deposit(msg.sender, _pid, _amount);
```

Figure 13 The source code of depositWithPid function

The *safeMdxTransfer* function will update the variable values of *govTotalSendProfit* and *userTotalSendProfit* to avoid repeated receipt of rewards; then call *withdrawPending* of the controller contract to issue rewards.

```
465     function safeMdxTransfer(address _token, address _to, uint256 _userPendingAmount, uint256 _govPendingAmount) private {
466         if(_userPendingAmount > 0 || _govPendingAmount > 0) {
467             govTotalSendProfit = govTotalSendProfit.add(_govPendingAmount);
468             userTotalSendProfit = userTotalSendProfit.add(_userPendingAmount);
469             IController(controller).withdrawPending(_token, _to, _userPendingAmount, _govPendingAmount);
470         }
471     }
```

Figure 14 The source code of *safeMdxTransfer* function

As shown in the figure below, the *earn* function calls the *approveCtr* function to approve tokens to the controller contract; then when the amount of tokens held by this contract is greater than the minimum earn amount of the corresponding pledge pool, the *earn* function of the controller contract is called to deposit.

```
479    function earn(address token) public {
480        PoolInfo storage pool = getPoolInfo(token);
481        // 授权
482        approveCtr(token);
483        if (IERC20(token).balanceOf(address(this)) > pool.earnLowerlimit) {
484            IController(controller).earn(token);
485        }
486    }
```

Figure 15 The source code of earn function

- Related functions: *depositWithPid*, *updatePool*, *safeMdxTransfer*, *earn*, *approveCtr*, *safeMdxTransfer*
- Safety Suggestion: None
- Result: Pass

3.3.6 Withdraw pledged tokens

- Description: The user can call the *withdraw* function to withdraw a specified number of pledged tokens; also call the *withdrawAll* function to extract all the tokens pledged by the caller; call the withdrawWithPid function to withdraw the pledged tokens of the specified pledge pool id.

The first two of the above functions mentioned will eventually call the *withdrawWithPid* function. The source code is as shown in the figure below, after checking the basic parameters, the function will call the *updatePool* function to update the corresponding pledge pool data; if the corresponding user has pledged, it will first settle the pledge rewards; if the number of tokens withdrawn this time is not 0, the corresponding token assets will be sent to the caller's address (the pledge quantity is 0, which means that only the reward is received); then, call the *earn* function to earn the excess tokens; finally update user and platform debt information.

```
399     function withdrawWithPid(uint256 _pid, uint256 _amount) public notPause {
400         require(_amount >= 0, "withdraw: not good");
401         PoolInfo storage pool = poolInfo[_pid];
402         UserInfo storage user = userInfo[_pid][msg.sender];
403         require(user.amount >= _amount, "withdraw: Insufficient balance");
404         updatePool(_pid);
405 
406         if(user.amount > 0){
407             // 给用户发放收益与平台分润
408             uint256 pendingAmount = user.amount.mul(pool.accMdxPerShare).div(1e12).sub(user.rewardDebt);
409             //uint256 pendingGovAmount = user.amount.mul(pool.govAccMdxPerShare).div(1e12).sub(user.govRewardDebt);
410             uint256 pendingGovAmount = govTotalProfit.sub(govTotalSendProfit);
411             safeMdxTransfer(address(pool.token), msg.sender, pendingAmount, pendingGovAmount);
412         }
413 
414         // 提现本金
415         if (_amount > 0) {
416             uint256 poolBalance = pool.token.balanceOf(address(this));
417             if(poolBalance < _amount) {
418                 // 当前合约余额不足，调用上游释放投资
419                 IController(controller).withdrawLp(address(pool.token), _amount.sub(poolBalance));
420                 poolBalance = pool.token.balanceOf(address(this));
421                 //上游资金不足 需要对冲
422                 require(poolBalance >= _amount, "withdraw: need hedge");
423             }
424 
425             user.amount = user.amount.sub(_amount);
426             pool.totalAmount = pool.totalAmount.sub(_amount);
427 
428             pool.token.safeTransfer(msg.sender, _amount);
429         }
430 
431         // 重新触发投资
432         earn(address(pool.token));
433 
434         // 更新用户负债
435         user.rewardDebt = user.amount.mul(pool.accMdxPerShare).div(1e12);
436         user.govRewardDebt = user.amount.mul(pool.govAccMdxPerShare).div(1e12);
437         emit Withdraw(msg.sender, _pid, _amount);
```

Figure 16 The source code of withdrawWithPid function

In addition, users can also call emergencyWithdraw function for emergency withdrawal. As shown in the figure below, this function does not issue pledge rewards, and directly refunds the full amount of pledged tokens (this means that pledge rewards will also be cleared).

```
441     function emergencyWithdraw(uint256 _pid) public notPause {
442         PoolInfo storage pool = poolInfo[_pid];
443         UserInfo storage user = userInfo[_pid][msg.sender];
444         uint256 amount = user.amount;
445 
446         uint256 poolBalance = pool.token.balanceOf(address(this));
447         if(poolBalance < amount){
448             // 当前合约余额不足，调用上游释放投资
449             IController(controller).withdrawLp(address(pool.token), amount.sub(poolBalance));
450 
451             poolBalance = pool.token.balanceOf(address(this));
452             // 上游资金不足 需要对冲
453             require(poolBalance >= amount, "withdraw: need hedge");
454         }
455 
456         user.amount = 0;
457         user.rewardDebt = 0;
458         user.govRewardDebt = 0;
459         pool.token.safeTransfer(msg.sender, amount);
460         pool.totalAmount = pool.totalAmount.sub(amount);
461         emit EmergencyWithdraw(msg.sender, _pid, amount);
462     }
```

Figure 17 The source code of emergencyWithdraw function

- Related functions: *withdraw, withdrawAll, withdrawWithPid, emergencyWithdraw, safeMdxTransfer*
- Safety Suggestion: None
- Result: Pass

3.4 The LiquidityOpt Contract Audit

3.4.1 Contract owner permission management

- Description: The highest permission owner of this contract (the contract deployer by default) can call the *transferOwnership* function to transfer the owner permission to the specified non-zero address; or call the *renounceOwnership* function to renounce the owner permission.
- Related functions: *transferOwnership, renounceOwnership,*
- Result: Pass

3.4.2 Set key parameters of the contract

- Description: The administrator of the contract can call the *setGovernance* function to set the Governance contract address; call the *setHubPool* function to set the HubPool address; call the *setRouter* function to set the router address.
- Related functions: *setGovernance, setHubPool, setRouterSafety*
- Suggestion: None
- Result: Pass

3.4.3 Administrator withdraws tokens

- Description: The contract administrator owner can call the *withdrawAll* and *withdrawToken* functions to withdraw the specified token, where the *withdrawToken* needs to specify the corresponding withdrawal amount.
- Related functions: withdrawAll, withdrawToken
- Safety Suggestion: None
- Result: Pass

3.4.4 Add and remove liquidity

- Description: The Governance contract can call the *liquidityDeposit* function of this contract to add liquidity. As shown in the figure below, the function will send tokens with added liquidity to the hubPool contract; if _triggerEarn is true, the *earn* function of the hubPool contract needs to be called to invest.

```
79      function liquidityDeposit(
80          address _token,
81          uint _amount,
82          bool _triggerEarn)
83      public onlyGovernance {
84          require(_amount > 0, "amount must > 0");
85
86          IERC20(_token).safeTransfer(hubPool, _amount);
87
88          uint index = IHubPool(hubPool).TokenOfPid(_token);
89          PoolInfo memory pool = IHubPool(hubPool).poolInfo(index);
90
91          if (_triggerEarn) {
92              IHubPool(hubPool).earn(address(pool.token));
93          }
94
95          emit LiquidityDepositSuccess(_amount);
96      }
```

Figure 18 The source code of liquidityDeposit function

The liquidityWithdraw function can be called by the Government contract to withdraw tokens. As shown in the figure below, this function will call the liquidityWithdraw function of the hubPool contract to transfer token assets to this contract.

```
98      function liquidityWithdraw(
99          address _token,
100         uint _amount)
101     internal onlyGovernance {
102         uint beforeBal = IERC20(_token).balanceOf(address(this));
103         IHubPool(hubPool).liquidityWithdraw(_token, _amount);
104         uint afterBal = IERC20(_token).balanceOf(address(this));
105
106         uint diff = afterBal.sub(beforeBal);
107         require(diff <= _amount, "withdraw inner error");
108
109         emit LiquidityWithdrawSuccess(diff);
110     }
```

Figure 19 The source code of liquidityWithdraw function

- Related functions: liquidityDeposit, liquidityWithdraw
- Safety Suggestion: None
- Result: Pass

3.4.5 Exchange tokens

- Description: The Governance contract can call the *withdrawAndSwap* function of this contract to withdraw A token; and exchange it for B token.

```
165     function withdrawAndSwap(
166         address _tokenA,
167         address _tokenB,
168         uint _amount,
169         uint _amountMin,
170         uint _deadline,
171         bool _triggerEarn,
172         uint _type)
173     external onlyGovernance returns (uint[] memory amounts) {
174         liquidityWithdraw(_tokenA, _amount);
175 
176         amounts = swapToken(_tokenA, _tokenB, _amount, _amountMin, _deadline, _triggerEarn, _type);
177     }
```

Figure 20 The source code of withdrawAndSwap function

As shown in the figure below, the *swapToken* function will first approve the exchange amount of A tokens to the Router contract; then, according to the input exchange type, splice the corresponding path, and call the token exchange function of the corresponding Router contract for token exchange; finally, the function The excess tokens will be returned to the fund pool.

```
120     internal onlyGovernance returns (uint[] memory amounts) {
121         require(router != address(0), "router address is zero");
122 
123         IERC20(_tokenA).approve(router, _amount);
124         address[] memory _tokenA2B = new address[](2);
125         // 把mdex换成 usdt
126         _tokenA2B[0] = _tokenA;
127         _tokenA2B[1] = _tokenB;
128 
129         if (_type == 1) {
130             // _amountExact => amountOutMin
131             amounts = IMdexRouter(router).swapExactTokensForTokens(
132                 _amount,
133                 _amountExact,
134                 _tokenA2B,
135                 address(this),
136                 _deadline
137             );
138         } else if (_type == 2) {
139             // _amountExact => amountInMax
140             amounts = IMdexRouter(router).swapTokensForExactTokens(
141                 _amount,
142                 _amountExact,
143                 _tokenA2B,
144                 address(this),
145                 _deadline
146             );
147         }
148 
149 
150         require(hubPool != address(0), "hubPool address is zero");
151 
152         // 将剩余的钱转回给资金池
153         uint bal = 0;
154         bal = balance(_tokenA, address(this));
155         if (bal > 0) {
156             liquidityDeposit(_tokenA, bal, _triggerEarn);
157         }
158         bal = balance(_tokenB, address(this));
159         if (bal > 0) {
160             liquidityDeposit(_tokenB, bal, _triggerEarn);
161         }
162     }
```

Figure 21 The source code of swapToken function

- Related functions: withdrawAndSwap, liquidityWithdraw, swapToken
- Safety Suggestion: None
- Result: Pass

3.5 The StrategyMdex Contract Audit

3.5.1 Contract owner permission management

- Description: The highest permission owner of this contract (the contract deployer by default) can call the *transferOwnership* function to transfer the owner permission to the specified non-zero address; or call the *renounceOwnership* function to renounce the owner permission; call the *setGovernance* function to set the Governance contract address.
- Related functions: *transferOwnership*, *renounceOwnership*, *setGovernance*

- Safety Suggestion: None
- Result: Pass

3.5.2 Set key parameters of the contract

- Description: The Governance contract can call related functions of this contract to set key parameters of the contract.
- Related functions: setGovernance, setController, setPid, setRouterAddr, setMdexTokenAddr, setSwapMiningAddr, setHecoPoolAddr
- Safety Suggestion: None
- Result: Pass

3.5.3 Add and remove liquidity

- Description: Any user can call the *addLiquidity* function to add liquidity. As shown in the figure below, this function will call the *approve* function of the corresponding token contract to approve tokens to the mdexRouterh contract; then call the *addLiquidity* function of the mdexRouterh contract to add the liquidity of this contract.

```
257     function addLiquidity(
258         uint amountADesired,
259         uint amountBDesired,
260         uint amountAMin,
261         uint amountBMin,
262         uint deadline
263     ) public returns (uint amountA, uint amountB, uint liquidity) {
264
265         // 从调用者转入 amountADesired 个 token0 到合约地址
266         // IERC20(tokenA).transferFrom(msg.sender, address(this), amountADesired);
267         // 给mdex router合约授权 amountADesired 个 token0
268         IERC20(tokenA).approve(mdexRouter, amountADesired);
269         // 从调用者转入 amountBDesired 个 tokenB 到合约地址
270         // IERC20(tokenB).transferFrom(msg.sender, address(this), amountBDesired);
271         // 给mdex router合约授权 amountBDesired 个 tokenB
272         IERC20(tokenB).approve(mdexRouter, amountBDesired);
273
274         // 调用 mdex 提供流动性
275         (uint amount0, uint amount1, uint lpAmount) = IMdexRouter(mdexRouter).addLiquidity(
276             tokenA,
277             tokenB,
278             amountADesired,
279             amountBDesired,
280             amountAMin,
281             amountBMin,
282             address(this),
283             deadline
284         );
285
286         return (amount0, amount1, lpAmount);
287     }
```

Figure 22 The source code of addLiquidity function

Similarly, the Governance contract can also call the removeLiquidity function to remove the liquidity of this contract.

```
290     function removeLiquidity(
291         uint liquidity,
292         uint amountAMin,
293         uint amountBMin,
294         uint deadline
295     ) public  returns (uint amountA, uint amountB) {
296         checkGovernance();
297         address _pair = IMdexFactory(factory()).pairFor(tokenA, tokenB);
298         // 给mdex router合约授权 amountBDesired 个 tokenB
299         IERC20(_pair).approve(mdexRouter, liquidity);
300 
301         // 调用 mdex 移除流动性
302         (uint amount0, uint amount1) = IMdexRouter(mdexRouter).removeLiquidity(
303             tokenA,
304             tokenB,
305             liquidity,
306             amountAMin,
307             amountBMin,
308             address(this),
309             deadline
310         );
311 
312         return (amount0, amount1);
313     }
```

Figure 23 The source code of removeLiquidity function

Among them, the *addLiquidity* function does not limit the caller of the function, which means that ordinary users can also directly call this function to add liquidity to the contract; but the removal can only be called by the Governance contract.

- Related functions: *addLiquidity*, *removeLiquidity*
- Safety Suggestion: None
- Result: Pass

3.6.4 Deposit and withdraw

- Description: The Governance contract can call the *depositStake* function of this contract to pledge tokens in the designated pledge pool; it can also call the *withdrawStake* function to withdraw the tokens pledged by the contract.
- Related functions: depositStak, withdrawStake
- Safety Suggestion: None
- Result: Pass

3.6.5 Withdraw liquidity assets

- Description: The Governance contract can call the *withdrawAll* function of this contract to withdraw the tokens that this contract has added to the liquidity pool to the vault contract. As shown in the figure below, the function will first call *withdrawMDXReward* to receive the reward of pledged LP tokens; then withdraw the pledged LP tokens, return the LP to the popularity pool, and withdraw the corresponding tokens; finally, send the withdrawn tokens to vault contract.

```
343     function withdrawAll() public  {
344         checkGovernance();
345         withdrawMDXReward();
346 
347         // 查看质押的LP数量
348         uint _lp_amount = balanceOf();
349         // 解除质押
350         if (_lp_amount > 0) {
351             withdrawStake(pid, _lp_amount);
352             removeLiquidity(_lp_amount, 0 ether, 0 ether, block.timestamp + 5 minutes);
353         }
354 
355         require(mdexTokenAddress != address(0), 'MDX == address(0)');
356 
357         // 获取金库地址
358         address vault_addr = Controller(controller).vaults();
359         // 将多余的token返还给金库
360         uint _after_bal0 = balance(tokenA, address(this));
361         uint _after_bal1 = balance(tokenB, address(this));
362         if (_after_bal0 > 0) {
363             IERC20(tokenA).transfer(vault_addr, _after_bal0);
364         }
365         if (_after_bal1 > 0) {
366             IERC20(tokenB).transfer(vault_addr, _after_bal1);
367         }
368 
369         uint u_balance = balance(mdexTokenAddress, address(this));
370         if (u_balance > 0) {
371             IERC20(mdexTokenAddress).transfer(vault_addr, u_balance);
372         }
373     }
```

Figure 24 The source code of withdrawAll function

- Related functions: *withdrawAll*, *withdrawMDXReward*
- Safety Suggestion: None
- Result: Pass

4. Conclusion

Beosin(ChengduLianAn) conducted a detailed audit on the design and code implementation of the smart contracts project coinwind-vault. All the issues found during the audit have been written into this audit report. Among them, the management permission owner and governance in the project have higher control rights over the entire project, and it is recommended to do a good job of permission control.The overall audit result of the smart contract project coinwind-vault is **Pass**.

Official Website

https://lianantech.com

E-mail

vaas@lianantech.com

Twitter

https://twitter.com/Beosin_com