



METATRUST

Pre Report for
Debox-box

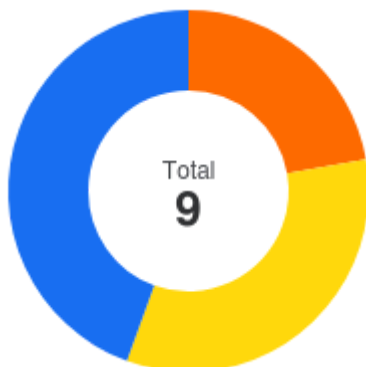
November 13, 2023






Executive Summary

| Overview | | | |
|--------------|---|--|--|
| Project Name | Debox-box | | |
| Codebase URL | Box (1).sol | | |
| Scan Engine | Security Analyzer | | |
| Scan Time | 2023/11/13 08:00:00 | | |
| Commit Id | 414396dad161ee5b454c21b5c6a279c299e8de31f2ba4bb0cedfcf79fd27e25 | | |

| Total | | | |
|----------------------|---|--|--|
| Critical Issues | 0 | | |
| High risk Issues | 2 | | |
| Medium risk Issues | 3 | | |
| Low risk Issues | 0 | | |
| Informational Issues | 4 | | |

| | | |
|---------------------|--|---|
| Critical Issues | | The issue can cause large economic losses, large-scale data disorder, loss of control of authority management, failure of key functions, or indirectly affect the correct operation of other smart contracts interacting with it. |
| High Risk Issues | | The issue puts a large number of users' sensitive information at risk or is reasonably likely to lead to catastrophic impacts on clients' reputations or serious financial implications for clients and users. |
| Medium Risk Issues | | The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact. |
| Low Risk Issues | | The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances. |
| Informational Issue | | The issue does not pose an immediate risk but is relevant to security best practices or Defence in Depth. |



| | | | |
|---|----------------------|-----|----------|
|  | Critical Issues | 0% | 0 |
|  | High risk Issues | 22% | 2 |
|  | Medium risk Issues | 33% | 3 |
|  | Low risk Issues | 0% | 0 |
|  | Informational Issues | 44% | 4 |

Summary of Findings

MetaScan security assessment was performed on **November 13, 2023 08:00:00** on project **Debox-box** with the repository on branch **default branch**. The assessment was carried out by scanning the project's codebase using the scan engine **Security Analyzer**. There are in total **9** vulnerabilities / security risks discovered during the scanning session, among which **0** critical vulnerabilities, **2** high risk vulnerabilities, **3** medium risk vulnerabilities, **0** low risk vulnerabilities, **4** informational issues.

| ID | Description | Severity | Alleviation |
|---------|--|---------------|--------------|
| MSA-001 | Possibility of lock ether when the sum of pre-sale meta boxes less than the <code>allocateBalance</code> | High risk | Fixed |
| MSA-002 | The locked ether caused by the <code>PER_BOX_GAS</code> part | High risk | Fixed |
| MSA-003 | Unable to transfer meta box | Medium risk | Acknowledged |
| MSA-004 | Centralization Risks | Medium risk | Acknowledged |
| MSA-005 | Unsafe usage of unchecked | Medium risk | |
| MSA-006 | Unused event | Informational | Fixed |
| MSA-007 | Missing Event Setter | Informational | Acknowledged |
| MSA-008 | Typos | Informational | Fixed |
| MSA-009 | The Price Model | Informational | Acknowledged |

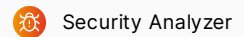
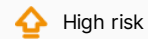
Findings

Critical (0)

No Critical vulnerabilities found here

High risk (2)

1. Possibility of lock ether when the sum of pre-sale meta boxes less than the `allocateBalance`



When users buy or sell meta boxes, the `getBuyAmount` function and the `getSellAmount` require the `_metaBoxes[meta].allocateBalance` is zero.

Let's consider this scenario:

- A meta box with a key "AAA" has a `allocateBalance` as 100;
- The "AAA" meta box only sold 80 meta boxes during the pre-sale;
- The owner allocates meta boxes to all the "AAA" meta box participants.

Now the "AAA" meta box's `allocateBalance` is `100 - 80`, 20, which is greater than 0.

As a result, the "AAA" meta box is unable to be traded, due to the `getBuyAmount` function and the `getSellAmount` requiring the `_metaBoxes[meta].allocateBalance` is zero when buying and selling, and the corresponding ether will be locked forever.

File(s) Affected

Box (1).sol #95-109

```
95     function start(bytes32 meta, bytes memory signature) external {
96         require(_metaBoxes[meta].expireTime == 0, "The box is started");
97         bytes32 message = keccak256(abi.encodePacked(msg.sender, meta));
98         require(_signOwner.isValidSignatureNow(message, signature), "The signature is invalid");
99         uint128 expireTime = uint128(block.timestamp+BOX_SALE_PERIOD);
100         _metaBoxes[meta] = BoxMeta({
101             owner: msg.sender,
102             expireTime: expireTime,
103             preSaleCnt:0,
104             tradeCnt:0,
105             index:0,
106             allocateBalance:uint128(BOX_SALE_CNT)
107         });
108         emit StartPreSale(msg.sender,meta,expireTime);
109     }
```

Box (1).sol #79-88

```
79     function getBuyAmount(bytes32 meta,uint128 cnt) public view returns(uint128,uint128){
80         require(_metaBoxes[meta].allocateBalance == 0,"The box is not start trade");
81         return _calculateTradeAmount(_metaBoxes[meta].tradeCnt,_metaBoxes[meta].tradeCnt+cnt);
82     }
83
84     function getSellAmount(bytes32 meta,uint128 cnt) public view returns(uint128,uint128){
85         require(_metaBoxes[meta].allocateBalance == 0,"The box is not start trade");
86         require(_metaBoxes[meta].tradeCnt >= cnt,"Insufficient box trade balance");
87         return _calculateTradeAmount(_metaBoxes[meta].tradeCnt-cnt,_metaBoxes[meta].tradeCnt);
88     }
```



Recommendation

Recommend adding logic to cover the case when the sold meta boxes during the pre-sale phase is less than the `allocateBalance`.

Alleviation Fixed

The team solved this issue by refunding users when a presale fails, in the new version smart contract whose sha256 value is 414396dadcd161ee5b454c21b5c6a279c299e8de31f2ba4bb0cedfcf79fd27e25.

2. The locked ether caused by the `PER_BOX_GAS` part

 High risk Security Analyzer

When users participate in a presale with the `preSale` function, users need to pay an amount of `box_cnt * PER_BOX_PRICE + PER_BOX_GAS` ether to join the presale, there are logics for refunding and selling boxes for the `box_cnt * PER_BOX_PRICE` part ether.

However, it seems to lack the logic to withdraw the `PER_BOX_GAS` part ether that is received in the `preSale` function, which results in accumulated ether being locked in the contract.

File(s) Affected

Box (1).sol #114-114

```
114         require(box_cnt * PER_BOX_PRICE + PER_BOX_GAS == msg.value , "Insufficient ether");
```

Recommendation


Recommend adding logic to process the `PER_BOX_GAS` part ether.

Alleviation Fixed

The team solved this issue by transferring `PER_BOX_GAS` part ether to `_signOwner`, in the new version smart contract whose sha256 value is 414396dadcd161ee5b454c21b5c6a279c299e8de31f2ba4bb0cedfcf79fd27e25.

Medium risk (3)

1. Unable to transfer meta box

 Medium risk Security Analyzer

Users can only buy and sell meta boxes with the updating of the `_metaBoxes` and `_userInfo`.

However, there is no related logic for users to transfer the meta boxes.

File(s) Affected

Box (1).sol #152-169

```
152     function buy(bytes32 meta,uint128 cnt) external payable nonReentrant {
153         require(cnt > 0,"Insufficient Box cnt");
154         (uint128 tradeTotalAmount, uint128 fee) = getBuyAmount(meta,cnt);
155         require(msg.value == tradeTotalAmount + fee ,"Insufficient pay amount");
156         _updateUserInfo(meta,cnt,true);
157         (uint128 ownerFees,uint128 farmAllFees,uint128 platfromFees) = _distributeFees(meta,fee);
158         emit Trade(msg.sender,meta,tradeTotalAmount,ownerFees,farmAllFees,platfromFees,true,cnt);
159     }
160
161     function sell(bytes32 meta,uint128 cnt) external nonReentrant {
162         require(cnt > 0,"Insufficient Box cnt");
163         require(_userInfo[meta][msg.sender].cnt >= cnt,"Insufficient Box balance");
164         (uint128 tradeTotalAmount, uint128 fee) = getSellAmount(meta,cnt);
165         _updateUserInfo(meta,cnt,false);
166         (uint128 ownerFees,uint128 farmAllFees,uint128 platfromFees) = _distributeFees(meta,fee);
167         _safeTransferEth(msg.sender,tradeTotalAmount - fee);
168         emit Trade(msg.sender,meta,tradeTotalAmount,ownerFees,farmAllFees,platfromFees,false,cnt);
169     }
```

Recommendation

Consider adding logic to transfer the meta boxes, meanwhile, process the `userInfo.claims` when transferring the meta boxes.

Alleviation Acknowledged

The team acknowledged this finding.

2. Centralization Risks



Medium risk



Security Analyzer

In the provided smart contract, here are the functions that to be centralized and owned by the contract owner:

- **setFeeReciever**(address fee_reciever): This function allows the owner to set the fee receiver address. The description could be: "The owner can set the fee receiver address, which is the address that will receive fees collected by the contract."
- **setBoxFee**(uint8 owner, uint8 platform, uint8 farm): This function enables the owner to set various fees related to box transactions. The description could be: "The owner can set the fees for different parties involved in box transactions, including the owner, platform, and farm."
- **allocate**(bytes32 meta, BoxAllocate[] memory box_allocates): This function allows the owner to allocate box purchases to users. The description could be: "The owner can allocate box purchases to users who participated in the pre-sale but were not allocated their boxes. This function helps distribute boxes to users who oversubscribed during the pre-sale."

File(s) Affected

Box (1).sol #66-77

```
66     function setFeeReciever (address fee_reciever) external onlyOwner {
67         require(address(0) != fee_reciever,"fee_reciever is zero addresss");
68         _feeReciever = fee_reciever;
69     }
70
71     function setBoxFee(uint8 owner,uint8 platform,uint8 farm) external onlyOwner {
72         owner_fee = owner;
73         platform_fee = platform;
74         farm_fee = farm;
75         trade_fee = owner_fee+platform_fee+farm_fee;
76         require(trade_fee <= 10,"over 10%");
77     }
```


Box (1).sol #123-150

```
123     function allocate(bytes32 meta,BoxAllocate[] memory box_allocates) external onlyOwner {
124         BoxMeta storage boxMeta = _metaBoxes[meta];
125         require(boxMeta.expireTime > 0 && boxMeta.expireTime < block.timestamp, "The box pre sale in g
126         BoxPreOrder[] storage boxPreOrders = _boxPreOrders[meta];
127         if (boxPreOrders.length >= box_allocates.length) {
128             uint128 allocateBalance = boxMeta.allocateBalance;
129             for (uint i = 0; i < box_allocates.length; i++) {
130                 BoxAllocate memory box_allocate = box_allocates[i];
131                 BoxPreOrder storage boxPreOrder = boxPreOrders[box_allocate.orderIndex];
132                 require(boxPreOrder.cnt > 0,"The box duplicate allocation");
133                 uint128 boxPreOrderCnt = boxPreOrder.cnt;
134                 require(box_allocate.cnt <= boxPreOrderCnt && box_allocate.cnt <= allocateBalance,"Insu
135                 boxPreOrder.cnt = 0;
136                 allocateBalance -= boxPreOrderCnt;
137                 if (box_allocate.cnt > 0) {
138                     _userInfo[meta][boxPreOrder.owner].cnt += box_allocate.cnt;
139                 }
140                 if (boxPreOrderCnt > box_allocate.cnt) {
141                     uint128 refund = (boxPreOrderCnt - box_allocate.cnt)*PER_BOX_PRICE;
142                     _safeTransferEth(boxPreOrder.owner,refund);
143                 }
144             }
145             boxMeta.allocateBalance = allocateBalance;
146         }
147         if (boxMeta.allocateBalance == 0 ) {
148             _metaBoxes[meta].tradeCnt = BOX_SALE_CNT;
149         }
150     }
```



Recommendation

Consider implementing a decentralized governance mechanism or a multi-signature scheme that requires consensus among multiple parties before pausing or unpausing the contract. This can help mitigate the centralization risk associated with a single owner controlling critical contract functions. Alternatively, you can provide a clear justification for the centralization aspect and ensure that users are aware of the potential risks associated with a single point of control.

Alleviation Acknowledged

The team acknowledged this finding.

3. Unsafe usage of unchecked

 Medium risk Security Analyzer

In the new version smart contract whose sha256 value is 414396dadcd161ee5b454c21b5c6a279c299e8de31f2ba4bb0cedfcf79fd27e25.

The `_calculateTradeAmount` function accumulates the total trade amount, and the `_calculateBoxPrice` function calculates the trade amount. There is a possibility for them to be overflowing, so, it is unsafe to use the `unchecked` block for them.

File(s) Affected

Box.sol #242-256

```
242     function _calculateTradeAmount(uint128 start_index, uint128 end_innex) internal view returns (uint128) {
243         require(end_innex > start_index, "Insufficient Box cnt");
244         uint128 tradeTotalAmount = 0;
245         unchecked {
246             for (uint128 i = end_innex; i > start_index; i--) {
247                 if (i > BOX_SALE_CNT) {
248                     tradeTotalAmount += _calculateBoxPrice(i);
249                 } else {
250                     tradeTotalAmount += PER_BOX_PRICE;
251                 }
252             }
253         }
254         uint128 fee = tradeTotalAmount * trade_fee / 100;
255         return (tradeTotalAmount, fee);
256     }
```

Box.sol #290-302

```
290     function _calculateBoxPrice(uint128 x) internal pure returns (uint128) {
291         if (x <= 100) return PER_BOX_PRICE;
292
293         uint128 boxPrice = 100;
294         unchecked {
295             if (x > 1000) {
296                 boxPrice = (x - 1000) ** 2 / 9 + 100 * (x - 100) / 3 + 100;
297             } else {
298                 boxPrice = 100 * (x - 100) / 3 + 100;
299             }
300         }
301         return boxPrice * 1e14;
302     }
```

Recommendation


Consider removing the usages of the `unchecked` block for the `_calculateTradeAmount` function and the `_calculateBoxPrice` function.

Low risk (0)

No Low risk vulnerabilities found here

Informational (4)

1. Unused event

 Informational Security Analyzer

The presence of an event that is declared but never used in the codebase. They may increase computation costs and lead to unnecessary gas consumption.

File(s) Affected

Box (1).sol #58-58

```
58      event StartTrade(address indexed sender, bytes32 meta);
```



Recommendation

Remove the unused event or emit it in the right place to avoid negative effects and improve code readability if there is no plan for further usage.

Alleviation Fixed

The team solved this issue by removing the redundant event, in the new version smart contract whose sha256 value is 414396dadc161ee5b454c21b5c6a279c299e8de31f2ba4bb0cedfcf79fd27e25.

2. Missing Event Setter

 Informational Security Analyzer

Functions update key states are recommended to emit the corresponding events.

File(s) Affected

Box (1).sol #66-69

```
66      function setFeeReciever (address fee_reciever) external onlyOwner {
67          require(address(0) != fee_reciever, "fee_reciever is zero addresss");
68          _feeReciever = fee_reciever;
69      }
```

Box (1).sol #71-77

```
71      function setBoxFee(uint8 owner,uint8 platform,uint8 farm) external onlyOwner {
72          owner_fee = owner;
73          platform_fee = platform;
74          farm_fee = farm;
75          trade_fee = owner_fee+platform_fee+farm_fee;
76          require(trade_fee <= 10, "over 10%");
77      }
```



Recommendation

Consider emitting the corresponding events.

Alleviation Acknowledged

The team acknowledged this issue.

3. Typos

 Informational Security Analyzer

The variable `toFram` is intended to be named `toFarm`.

The variable `platfromFees` is intended to be named `platformFees`.

File(s) Affected

Box (1).sol #166-168

```
166         (uint128 ownerFees,uint128 farmAllFees,uint128 platfromFees) = _distributeFees(meta,fee);
167         _safeTransferEth(msg.sender,tradeTotalAmount - fee);
168         emit Trade(msg.sender,meta,tradeTotalAmount,ownerFees,farmAllFees,platfromFees,false,cnt);
```

Box (1).sol #59-59

```
59         event Trade(address indexed sender, bytes32 meta,uint128 amount,uint128 ownerFees,uint128 farmAllFee
```

Box (1).sol #194-206

```
194         function _distributeFees(bytes32 meta, uint128 fees) internal returns (uint128 toOwner,uint128 toF
195             uint128 base=trade_fee;
196             if (base==0) return (0,0,0);
197             BoxMeta storage boxMeta = _metaBoxes[meta];
198             unchecked{
199                 toOwner= fees*owner_fee/base;
200                 if(boxMeta.tradeCnt>0) toFram = fees*farm_fee/base;
201                 toCore = fees - toOwner - toFram;
202             }
203             if (toFram>0) boxMeta.index += toFram/boxMeta.tradeCnt;
204             if (toOwner>0) _safeTransferEth(boxMeta.owner,toOwner);
205             if (toCore>0)_safeTransferEth(_feeReciever,toCore);
206         }
```

Box (1).sol #157-158

```
157         (uint128 ownerFees,uint128 farmAllFees,uint128 platfromFees) = _distributeFees(meta,fee);
158         emit Trade(msg.sender,meta,tradeTotalAmount,ownerFees,farmAllFees,platfromFees,true,cnt);
```



Recommendation

Recommend updating these typos.

Alleviation Fixed

The team solved this issue by correcting the typos, in the new version smart contract whose sha256 value is 414396dadc161ee5b454c21b5c6a279c299e8de31f2ba4bb0cedfcf79fd27e25.

4. The Price Model

 Informational Security Analyzer

The price model designed in the `_calculateBoxPrice` function calculates the price with three formulas,

```
function _calculateBoxPrice(uint128 x) internal pure returns (uint128) {
    if (x <= 100) return PER_BOX_PRICE;

    uint128 boxPrice = 100;
    unchecked {
        if (x > 1000) {
            boxPrice = (x - 1000) ** 2 / 9 + 100 * (x - 100) / 3 + 100;
        } else {
            boxPrice = 100 * (x - 100) / 3 + 100;
        }
    }
    return boxPrice * 1e14;
}
```

It may incur great slippage, especially when the variable `x` is greater than 1000.

File(s) Affected

Box (1).sol #225-233

```
225     function _calculateBoxPrice(uint128 x) internal pure returns (uint128) {
226         uint128 boxPrice = 100;
227         if (x > 1000) {
228             boxPrice = (x - 1000)**2 / 9 + 100*(x - 100)/3 + 100;
229         }else if (x > 100) {
230             boxPrice = 100 * (x - 100) / 3 + 100;
231         }
232         return boxPrice*10**14;
233     }
```

Recommendation

Consider checking if the price model is an intended design.

Alleviation Acknowledged

The team acknowledged this issue.

Audit Scope

| File | SHA256 | File Path |
|-------------|--|--------------|
| Box (1).sol | f581723b3cb6f4c3ae5977ae64fd7f09b8734e93656778019e814bbe2cb7f0a0 | /Box (1).sol |

Disclaimer

This report is governed by the stipulations (including but not limited to service descriptions, confidentiality, disclaimers, and liability limitations) outlined in the Services Agreement, or as detailed in the scope of services and terms provided to you, the Customer or Company, within the context of the Agreement. The Company is permitted to use this report only as allowed under the terms of the Agreement. Without explicit written permission from MetaTrust, this report must not be shared, disclosed, referenced, or depended upon by any third parties, nor should copies be distributed to anyone other than the Company.

It is important to clarify that this report neither endorses nor disapproves any specific project or team. It should not be viewed as a reflection of the economic value or potential of any product or asset developed by teams or projects engaging MetaTrust for security evaluations. This report does not guarantee that the technology assessed is completely free of bugs, nor does it comment on the business practices, models, or legal compliance of the technology's creators.

This report is not intended to serve as investment advice or a tool for investment decisions related to any project. It represents a thorough assessment process aimed at enhancing code quality and mitigating risks inherent in cryptographic tokens and blockchain technology. Blockchain and cryptographic assets inherently carry ongoing risks. MetaTrust's role is to support companies and individuals in their security diligence and to reduce risks associated with the use of emerging and evolving technologies. However, MetaTrust does not guarantee the security or functionality of the technologies it evaluates.

MetaTrust's assessment services are contingent on various dependencies and are continuously evolving. Accessing or using these services, including reports and materials, is at your own risk, on an as-is and as-available basis. Cryptographic tokens are novel technologies with inherent technical risks and uncertainties. The assessment reports may contain inaccuracies, such as false positives or negatives, and unpredictable outcomes. The services may rely on multiple third-party layers.

All services, labels, assessment reports, work products, and other materials, or any results from their use, are provided "as is" and "as available," with all faults and defects, without any warranty. MetaTrust expressly disclaims all warranties, whether express, implied, statutory, or otherwise, including but not limited to warranties of merchantability, fitness for a particular purpose, title, non-infringement, and any warranties arising from course of dealing, usage, or trade practice. MetaTrust does not guarantee that the services, reports, or materials will meet specific requirements, be error-free, or be compatible with other software, systems, or services.

Neither MetaTrust nor its agents make any representations or warranties regarding the accuracy, reliability, or currency of any content provided through the services. MetaTrust is not liable for any content inaccuracies, personal injuries, property damages, or any loss resulting from the use of the services, reports, or materials.

Third-party materials are provided "as is," and any warranty concerning them is strictly between the Customer and the third-party owner or distributor. The services, reports, and materials are intended solely for the Customer and should not be relied upon by others or shared without MetaTrust's consent. No third party or representative thereof shall have any rights or claims against MetaTrust regarding these services, reports, or materials.

The provisions and warranties of MetaTrust in this agreement are exclusively for the Customer's benefit. No third party has any rights or claims against MetaTrust regarding these provisions or warranties. For clarity, the services, including any assessment reports or materials, should not be used as financial, tax, legal, regulatory, or other forms of advice.