METATRUST

Pre Report for

Debox-box

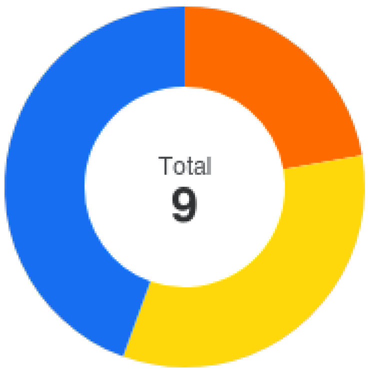
November 13, 2023

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	414396dadc161ee5b454c21b5c6a279c299e8de31f2ba4bb0cedfcf79fd27e25

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.

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



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 `allocateBalance`	High risk	Fixed
MSA-002	The locked ether caused by the `PER_BOX_GAS` 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`

High risk Security Analyzer

When users buy or sell meta boxes, the `getBuyAmount` function and the `getSellAmount` require the `_metaBoxs[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 `_metaBoxs[meta].allocateBalance` is zero when buying and selling, and the corresponding ether will be locked forever.

File(s) Affected

Box (1).sol #95-109

```
    function start(bytes32 meta, bytes memory signature) external {
        require(_metaBoxs[meta].expireTime == 0, "The box is started");
        bytes32 message = keccak256(abi.encodePacked(msg.sender, meta));
        require(_signOwner.isValidSignatureNow(message,signature), "The signature is invalid");
        uint128 expireTime = uint128(block.timestamp+BOX_SALE_PERIOD);
        _metaBoxs[meta] = BoxMeta({
                owner: msg.sender,
                expireTime: expireTime,
                preSaleCnt:0,
                tradeCnt:0,
                index:0,
                allocateBalance:uint128(BOX_SALE_CNT)
        });
        emit StartPreSale(msg.sender,meta,expireTime);
    }
```

Box (1).sol #79-88

```
    function getBuyAmount(bytes32 meta,uint128 cnt) public view returns(uint128,uint128){
        require(_metaBoxs[meta].allocateBalance == 0,"The box is not start trade");
        return _calculateTradeAmount(_metaBoxs[meta].tradeCnt,_metaBoxs[meta].tradeCnt+cnt);
    }

    function getSellAmount(bytes32 meta,uint128 cnt) public view returns(uint128,uint128){
        require(_metaBoxs[meta].allocateBalance == 0,"The box is not start trade");
        require(_metaBoxs[meta].tradeCnt >= cnt,"Insufficient box trade balance");
        return _calculateTradeAmount(_metaBoxs[meta].tradeCnt-cnt,_metaBoxs[meta].tradeCnt);
    }
```

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 414396dadc161ee5b454c21b5c6a279c299e8de31f2ba4bb0cedfcf79fd27e25.

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

```
        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 414396dadc161ee5b454c21b5c6a279c299e8de31f2ba4bb0cedfcf79fd27e25.

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 `_metaBoxs` and `_userInfo`.

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

File(s) Affected

Box (1).sol #152-169

```
    function buy(bytes32 meta,uint128 cnt) external payable nonReentrant {
        require(cnt > 0,"Insufficient Box cnt");
        (uint128 tradeTotalAmount, uint128 fee)  = getBuyAmount(meta,cnt);
        require(msg.value == tradeTotalAmount + fee ,"Insufficient pay amount");
        _updateUserInfo(meta,cnt,true);
        (uint128 ownerFees,uint128 farmAllFees,uint128 platfromFees) = _distributeFees(meta,fee);
        emit Trade(msg.sender,meta,tradeTotalAmount,ownerFees,farmAllFees,platfromFees,true,cnt);
    }

    function sell(bytes32 meta,uint128 cnt) external nonReentrant {
        require(cnt > 0,"Insufficient Box cnt");
        require(_userInfo[meta][msg.sender].cnt >= cnt,"Insufficient Box balance");
        (uint128 tradeTotalAmount, uint128 fee)  = getSellAmount(meta,cnt);
        _updateUserInfo(meta,cnt,false);
        (uint128 ownerFees,uint128 farmAllFees,uint128 platfromFees) = _distributeFees(meta,fee);
        _safeTransferEth(msg.sender,tradeTotalAmount - fee);
        emit Trade(msg.sender,meta,tradeTotalAmount,ownerFees,farmAllFees,platfromFees,false,cnt);
    }
```

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

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

    function setBoxFee(uint8 owner,uint8 platform,uint8 farm) external onlyOwner {
        owner_fee = owner;
        platform_fee = platform;
        farm_fee = farm;
        trade_fee = owner_fee+platform_fee+farm_fee;
        require(trade_fee <= 10,"over 10%");
    }
```

Box (1).sol #123-150

```
    function allocate(bytes32 meta,BoxAllocate[] memory box_allocates) external onlyOwner {
        BoxMeta storage boxMeta = _metaBoxs[meta];
        require(boxMeta.expireTime > 0  && boxMeta.expireTime < block.timestamp, "The box pre sale in p
        BoxPreOrder[] storage boxPreOrders = _boxPreOrders[meta];
        if (boxPreOrders.length >= box_allocates.length) {
            uint128 allocateBalance = boxMeta.allocateBalance;
            for (uint i = 0; i < box_allocates.length; i++) {
                BoxAllocate memory box_allocate = box_allocates[i];
                BoxPreOrder storage boxPreOrder = boxPreOrders[box_allocate.orderIndex];
                require(boxPreOrder.cnt > 0,"The box duplicate allocation");
                uint128 boxPreOrderCnt = boxPreOrder.cnt;
                require(box_allocate.cnt <= boxPreOrderCnt && box_allocate.cnt <= allocateBalance,"Insu
                boxPreOrder.cnt = 0;
                allocateBalance -= boxPreOrderCnt;
                if (box_allocate.cnt > 0) {
                    _userInfo[meta][boxPreOrder.owner].cnt +=  box_allocate.cnt;
                }
                if (boxPreOrderCnt > box_allocate.cnt) {
                    uint128 refound = (boxPreOrderCnt - box_allocate.cnt)*PER_BOX_PRICE;
                    _safeTransferEth(boxPreOrder.owner,refound);
                }
            }
            boxMeta.allocateBalance = allocateBalance;
        }
        if (boxMeta.allocateBalance == 0 ) {
            _metaBoxs[meta].tradeCnt = BOX_SALE_CNT;
        }
    }
```

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 414396dadc161ee5b454c21b5c6a279c299e8de31f2ba4bb0cedfcf79fd27e25.

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

```
    function _calculateTradeAmount(uint128 start_index, uint128 end_innex) internal view returns (uint1
        require(end_innex > start_index, "Insufficient Box cnt");
        uint128 tradeTotalAmount = 0;
        unchecked {
            for (uint128 i = end_innex; i > start_index; i--) {
                if (i > BOX_SALE_CNT) {
                    tradeTotalAmount += _calculateBoxPrice(i);
                } else {
                    tradeTotalAmount += PER_BOX_PRICE;
                }
            }
        }
        uint128 fee = tradeTotalAmount * trade_fee / 100;
        return (tradeTotalAmount, fee);
    }
```

Box.sol #290-302

```
    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;
    }
```

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

```
    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

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

Box (1).sol #71-77

```
    function setBoxFee(uint8 owner,uint8 platform,uint8 farm) external onlyOwner {
        owner_fee = owner;
        platform_fee = platform;
        farm_fee = farm;
        trade_fee = owner_fee+platform_fee+farm_fee;
        require(trade_fee <= 10,"over 10%");
    }
```

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

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

Box (1).sol #59-59

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

Box (1).sol #194-206

```
    function _distributeFees(bytes32 meta, uint128 fees) internal returns (uint128 toOwner,uint128 toF
        uint128 base=trade_fee;
      if (base==0) return(0,0,0);
        BoxMeta storage boxMeta = _metaBoxs[meta];
        unchecked{
            toOwner=  fees*owner_fee/base;
          if(boxMeta.tradeCnt>0)  toFram =  fees*farm_fee/base;
          toCore =  fees - toOwner - toFram;
        }
        if (toFram>0) boxMeta.index += toFram/boxMeta.tradeCnt;
        if (toOwner>0) _safeTransferEth(boxMeta.owner,toOwner);
        if (toCore>0)_safeTransferEth(_feeReciever,toCore);
    }
```

Box (1).sol #157-158

```
        (uint128 ownerFees,uint128 farmAllFees,uint128 platfromFees) = _distributeFees(meta,fee);
        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

```
    function _calculateBoxPrice(uint128 x) internal pure returns (uint128) {
        uint128 boxPrice = 100;
        if (x > 1000) {
            boxPrice = (x - 1000)**2 / 9 + 100*(x - 100)/3 + 100;
        }else if (x > 100) {
            boxPrice = 100 * (x - 100) / 3 + 100;
        }
        return boxPrice*10**14;
    }
```

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	f581723b3cb6f4c3ae5977ae64fd7f09b8734e9365677 8019e814bbe2cb7f0a0	/Box (1).sol

Disclaimer

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.