Smart Contract Security Audit Report

[2021]

Table Of Contents

1 Executive Summary

On 2021.06.21, the SlowMist security team received the DODO team's security audit application for DODO NFT DropsV2, DODOMiningV3, DODO ERC20 FactoryV2, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

Level	Description
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.

Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

- Reentrancy Vulnerability
- Replay Vulnerability
- Reordering Vulnerability
- Short Address Vulnerability
- Denial of Service Vulnerability
- Transaction Ordering Dependence Vulnerability
- Race Conditions Vulnerability
- Authority Control Vulnerability
- Integer Overflow and Underflow Vulnerability
- TimeStamp Dependence Vulnerability
- Uninitialized Storage Pointers Vulnerability
- Arithmetic Accuracy Deviation Vulnerability
- tx.origin Authentication Vulnerability
- "False top-up" Vulnerability

- Variable Coverage Vulnerability
- Gas Optimization Audit
- Malicious Event Log Audit
- Redundant Fallback Function Audit
- Unsafe External Call Audit
- Explicit Visibility of Functions State Variables Aduit
- Design Logic Audit
- Scoping and Declarations Audit

3 Project Overview

3.1 Project Introduction

Audit version code：

Working scope 1：

https://github.com/DODOEX/contractV2/tree/feature/nft

commit: 453e323

- contracts/DODODrops/DODODropsV2/
- contracts/SmartRoute/proxies/DODODropsProxy.sol

Working scope 2：

https://github.com/DODOEX/contractV2/tree/feature/mineUpdate

commit: c7202ee

- contracts/DODOToken/DODOMineV3/
- contracts/Factory/Registries/DODOMineV3Registry.sol

- contracts/SmartRoute/proxies/DODOMineV3Proxy.sol
- contracts/external/ERC20/CustomERC20.sol
- contracts/Factory/ERC20V2Factory.sol
- contracts/SmartRoute/proxies/DODORouteProxy.sol

Fix version code： The issues found in the audit were fixed in the following commit.

https://github.com/DODOEX/contractV2/commit/fe8c2ba7e9b6e91063830f225f8a5cc00cb4223c

https://github.com/DODOEX/contractV2/commit/0a574fff8da791061b6b12c14bda7d669bc0812d

https://github.com/DODOEX/contractV2/commit/7e629d0e58ac50a19ce476f529cfa213f9994715

https://github.com/DODOEX/contractV2/commit/7acd630e506bf394c4c93e61cdc5b025357a5d5d

https://github.com/DODOEX/contractV2/commit/5cda7ef2a3746e9025f4310e7bf830d5e1f4a0bf

https://github.com/DODOEX/contractV2/commit/b0e91d2a092c6994a420b375a680a123af498e52

https://github.com/DODOEX/contractV2/commit/eed61b50d4df52d07717368b70d094a6d10f73c7

3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	Random can be predicted issue	Design Logic Audit	Critical	Confirmed
N2	isContract can be bypassed	Design Logic Audit	Medium	Fixed
N3	Excessive authority issue	Authority Control Vulnerability	Medium	Confirmed
N4	The DoS risk	Others	Suggestion	Confirmed
N5	Event log missing	Others	Suggestion	Confirmed
N6	Check enhancement of isLpToken	Design Logic Audit	Suggestion	Confirmed

NO	Title	Category	Level	Status
N7	Security reminder on architecture design	Others	Suggestion	Confirmed

4 Code Overview

4.1 Contracts Description

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

CustomERC20			
Function Name	Visibility	Mutability	Modifiers
init	Public	Can Modify State	-
transfer	Public	Can Modify State	-
balanceOf	Public	-	-
transferFrom	Public	Can Modify State	-
approve	Public	Can Modify State	-
allowance	Public	-	-
_transfer	Internal	Can Modify State	-
mint	External	Can Modify State	onlyOwner

CustomERC20			
burn	External	Can Modify State	onlyOwner
changeTeamAccount	External	Can Modify State	onlyOwner

DODODropsProxy			
Function Name	Visibility	Mutability	Modifiers
<Fallback>	External	Payable	-
<Receive Ether>	External	Payable	-
<Constructor>	Public	Can Modify State	-
buyTickets	External	Payable	preventReentrant

DODODrops			
Function Name	Visibility	Mutability	Modifiers
<Fallback>	External	Payable	-
<Receive Ether>	External	Payable	-
init	Public	Can Modify State	-
buyTickets	External	Payable	preventReentrant
redeemTicket	External	Can Modify State	-
_redeemSinglePrize	Internal	Can Modify State	-
_setSellingInfo	Internal	Can Modify State	-
_setProbInfo	Internal	Can Modify State	-
_setFixedAmountInfo	Internal	Can Modify State	-

DODODrops			
withdraw	External	Can Modify State	onlyOwner
setRevealRn	External	Can Modify State	onlyOwner
setSellingInfo	External	Can Modify State	notStart onlyOwner
setProbInfo	External	Can Modify State	notStart onlyOwner
setFixedAmountInfo	External	Can Modify State	notStart onlyOwner
addFixedAmountInfo	External	Can Modify State	notStart onlyOwner
setTokenIdMapByIndex	External	Can Modify State	notStart onlyOwner
updateRNG	External	Can Modify State	onlyOwner
updateTicketUnit	External	Can Modify State	onlyOwner
updateRedeemTime	External	Can Modify State	onlyOwner
getSellingStage	Public	-	-
getSellingInfo	Public	-	-
addressToShortString	Public	-	-

ERC721			
Function Name	Visibility	Mutability	Modifiers
supportsInterface	Public	-	-
balanceOf	Public	-	-
ownerOf	Public	-	-
name	Public	-	-

ERC721			
symbol	Public	-	-
tokenURI	Public	-	-
_baseURI	Internal	-	-
approve	Public	Can Modify State	-
getApproved	Public	-	-
setApprovalForAll	Public	Can Modify State	-
isApprovedForAll	Public	-	-
transferFrom	Public	Can Modify State	-
safeTransferFrom	Public	Can Modify State	-
safeTransferFrom	Public	Can Modify State	-
_safeTransfer	Internal	Can Modify State	-
_exists	Internal	-	-
_isApprovedOrOwner	Internal	-	-
_safeMint	Internal	Can Modify State	-
_safeMint	Internal	Can Modify State	-
_mint	Internal	Can Modify State	-
_burn	Internal	Can Modify State	-
_transfer	Internal	Can Modify State	-
_approve	Internal	Can Modify State	-
_checkOnERC721Received	Private	Can Modify State	-

ERC721			
_beforeTokenTransfer	Internal	Can Modify State	-

ERC721Enumerable			
Function Name	Visibility	Mutability	Modifiers
supportsInterface	Public	-	-
tokenOfOwnerByIndex	Public	-	-
totalSupply	Public	-	-
tokenByIndex	Public	-	-
_beforeTokenTransfer	Internal	Can Modify State	-
_addTokenToOwnerEnumeration	Private	Can Modify State	-
_addTokenToAllTokensEnumeration	Private	Can Modify State	-
_removeTokenFromOwnerEnumeration	Private	Can Modify State	-
_removeTokenFromAllTokensEnumeration	Private	Can Modify State	-

DropsERC721			
Function Name	Visibility	Mutability	Modifiers
addMintAccount	Public	Can Modify State	onlyOwner
removeMintAccount	Public	Can Modify State	onlyOwner
init	Public	Can Modify State	-
mint	External	Can Modify State	-

ERC1155			
Function Name	Visibility	Mutability	Modifiers
supportsInterface	Public	-	-
uri	Public	-	-
balanceOf	Public	-	-
balanceOfBatch	Public	-	-
setApprovalForAll	Public	Can Modify State	-
isApprovedForAll	Public	-	-
safeTransferFrom	Public	Can Modify State	-
safeBatchTransferFrom	Public	Can Modify State	-
_setURI	Internal	Can Modify State	-
_mint	Internal	Can Modify State	-
_mintBatch	Internal	Can Modify State	-
_burn	Internal	Can Modify State	-
_burnBatch	Internal	Can Modify State	-
_beforeTokenTransfer	Internal	Can Modify State	-
_doSafeTransferAcceptanceCheck	Private	Can Modify State	-
_doSafeBatchTransferAcceptanceCheck	Private	Can Modify State	-
_asSingletonArray	Private	-	-

DropsERC1155

DropsERC1155			
Function Name	Visibility	Mutability	Modifiers
addMintAccount	Public	Can Modify State	onlyOwner
removeMintAccount	Public	Can Modify State	onlyOwner
init	Public	Can Modify State	-
mint	External	Can Modify State	-
uri	Public	-	-

DropsFeeModel			
Function Name	Visibility	Mutability	Modifiers
addDropBoxInfo	External	Can Modify State	onlyOwner
setDropBoxInfo	External	Can Modify State	onlyOwner
getPayAmount	External	-	-

BaseMine			
Function Name	Visibility	Mutability	Modifiers
getPendingReward	Public	-	-
getPendingRewardByToken	External	-	-
totalSupply	Public	-	-
balanceOf	Public	-	-
getRewardTokenById	External	-	-
getIdByRewardToken	Public	-	-

BaseMine			
getRewardNum	External	-	-
getVaultByRewardToken	Public	-	-
getVaultDebtByRewardToken	Public	-	-
claimReward	Public	Can Modify State	-
claimAllRewards	External	Can Modify State	-
addRewardToken	External	Can Modify State	onlyOwner
setEndBlock	External	Can Modify State	onlyOwner
setReward	External	Can Modify State	onlyOwner
withdrawLeftOver	External	Can Modify State	onlyOwner
directTransferOwnership	External	Can Modify State	onlyOwner
_updateReward	Internal	Can Modify State	-
_updateAllReward	Internal	Can Modify State	-
_getUnrewardBlockNum	Internal	-	-
_getAccRewardPerShare	Internal	-	-

ERC20Mine			
Function Name	Visibility	Mutability	Modifiers
init	External	Can Modify State	-
deposit	External	Can Modify State	-
withdraw	External	Can Modify State	-

RewardVault			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
reward	External	Can Modify State	onlyOwner
withdrawLeftOver	External	Can Modify State	onlyOwner
syncValue	External	Can Modify State	-

DODOMineV3Proxy			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
createDODOMineV3	External	Can Modify State	-
depositRewardToVault	External	Can Modify State	-
depositRewardToMine	External	Can Modify State	-
updateMineV2Template	External	Can Modify State	onlyOwner

DODOMineV3Registry			
Function Name	Visibility	Mutability	Modifiers
addMineV3	External	Can Modify State	-
removeMineV3	External	Can Modify State	onlyOwner
addAdminList	External	Can Modify State	onlyOwner
removeAdminList	External	Can Modify State	onlyOwner
addSingleTokenList	External	Can Modify State	onlyOwner

DODOMineV3Registry			
removeSingleTokenList	External	Can Modify State	onlyOwner

DODORouteProxy			
Function Name	Visibility	Mutability	Modifiers
<Fallback>	External	Payable	-
<Receive Ether>	External	Payable	-
<Constructor>	Public	Can Modify State	-
mixSwap	External	Payable	judgeExpired
dodoMutliSwap	External	Payable	judgeExpired
_multiSwap	Internal	Can Modify State	-
_deposit	Internal	Can Modify State	-

ERC20Factory			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
createStdERC20	External	Can Modify State	-
createMintableERC20	External	Can Modify State	-
getTokenByUser	External	-	-

4.3 Vulnerability Summary

[N1] [Critical] Random can be predicted issue

Category: Design Logic Audit

Content

The random number is uncertain when buying a ticket. However, there is no separate operation of using the redeeming tickets and determining the random number when redeeming tickets, and there is an issue that the random number can be predicted.

- https://github.com/DODOEX/contractV2/blob/453e323af6/contracts/DODODrops/DODODropsV2/DODODrops.sol#L114

```
function buyTickets(address ticketTo, uint256 ticketAmount) payable external
preventReentrant {
    (uint256 curPrice, uint256 sellAmount, uint256 index) = getSellingInfo();
    require(curPrice > 0 && sellAmount > 0, "CAN_NOT_BUY");
    require(ticketAmount <= sellAmount, "TICKETS_NOT_ENOUGH");
    (uint256 payAmount, uint256 feeAmount) =
IDropsFeeModel(_FEE_MODEL_).getPayAmount(address(this), ticketTo, curPrice,
ticketAmount);
    require(payAmount > 0, "UnQualified");
    uint256 baseBalance = IERC20(_BUY_TOKEN_).universalBalanceOf(address(this));
    uint256 buyInput = baseBalance.sub(_BUY_TOKEN_RESERVE_);
    require(payAmount <= buyInput, "PAY_AMOUNT_NOT_ENOUGH");
    _SELLING_AMOUNT_SET_[index] = sellAmount.sub(ticketAmount);
    _BUY_TOKEN_RESERVE_ = baseBalance.sub(feeAmount);
    IERC20(_BUY_TOKEN_).universalTransfer(_MAINTAINER_,feeAmount);
    _mint(ticketTo, ticketAmount);
    emit BuyTicket(ticketTo, payAmount, feeAmount, ticketAmount);
}
```

The owner determines the value of `_REVEAL_RN_` by calling the setRevealRn function. The value of `_REVEAL_RN_` will affect the result of the random number.

- https://github.com/DODOEX/contractV2/blob/453e323af6/contracts/DODODrops/DODODropsV2/DODODrops.sol#L229

```
function setRevealRn() external onlyOwner {
    require(_REVEAL_RN_ == 0, "ALREADY_SET");
```

```
        _REVEAL_RN_ = uint256(keccak256(abi.encodePacked(blockhash(block.number - 1))));
        emit SetReveal();
    }
```

The value of `random` is related to `_REVEAL_RN_`, `msg.sender`, `balanceOf(msg.sender)` and `curNo` in REVEAL_MODE mode when users use wallets for transactions. Attackers can generate addresses and balances values to control the random number. In non-REVEAL_MODE mode, the value of `random` is related to `_RNG_`, `block.number`, and `gasleft`. The attackers can sort transactions through pre-execution or in cooperation with miners. In this way, they can manipulate block.number and gasleft to control random numbers.

- https://github.com/DODOEX/contractV2/blob/453e323af6/contracts/DODODrops/DODODropsV2/DODODrops.sol#L154-L159

```
 function _redeemSinglePrize(address to, uint256 curNo, address referer) internal {
        require(block.timestamp >= _REDEEM_ALLOWED_TIME_ && _REDEEM_ALLOWED_TIME_ !=
0, "REDEEM_CLOSE");
        uint256 range;
        if(_IS_PROB_MODE_) {
            range = _PROB_INTERVAL_[_PROB_INTERVAL_.length - 1];
        }else {
            range = _TOKEN_ID_LIST_.length;
        }
        uint256 random;
        if(_IS_REVEAL_MODE_) {
            require(_REVEAL_RN_ != 0, "REVEAL_NOT_SET");
            random = uint256(keccak256(abi.encodePacked(_REVEAL_RN_, msg.sender,
balanceOf(msg.sender).add(curNo + 1)))) % range;
        }else {
            random = IRandomGenerator(_RNG_).random(gasleft() + block.number) %
range;
        }
        uint256 tokenId;
        if(_IS_PROB_MODE_) {
            uint256 i;
            for (i = 0; i < _PROB_INTERVAL_.length; i++) {
                if (random <= _PROB_INTERVAL_[i]) {
                    break;
                }
            }
            require(_TOKEN_ID_MAP_[i].length > 0, "EMPTY_TOKEN_ID_MAP");
            tokenId = _TOKEN_ID_MAP_[i][random % _TOKEN_ID_MAP_[i].length];
            IDropsNft(_NFT_TOKEN_).mint(to, tokenId, 1, "");
        } else {
            tokenId = _TOKEN_ID_LIST_[random];
            if(random != range - 1) {
                _TOKEN_ID_LIST_[random] = _TOKEN_ID_LIST_[range - 1];
            }
            _TOKEN_ID_LIST_.pop();
            IDropsNft(_NFT_TOKEN_).mint(to, tokenId);
        }
        emit RedeemPrize(to, tokenId, referer);
    }
```

Solution

It is recommended that the operation of redeeming bills and determining the random number be implemented in a two-step call, and the random number uses the value of the future block hash or chainlink as the source of the random number seed.

Status

Confirmed; The IS_REVEAL_MODE mode issue was fixed in commit: fe8c2ba7e9b6e91063830f225f8a5cc00cb4223c by restricting the transaction of the ticket. The non-IS_REVEAL_MODE mode restricts only EOA users can participate, and the attacker can sort the transactions through pre-execution or cooperation with miners to manipulate the results of random numbers.

[N2] [Medium] isContract can be bypassed

Category: Design Logic Audit

Content

When redeeming tickets isContract is used to determine whether the caller msg.sender is a contract. The contract is not allowed to be called, but the implementation of this check has flaws and can be bypassed.

- https://github.com/DODOEX/contractV2/blob/453e323af6/contracts/DODODrops/DODODropsV2/DODODrops.sol#L135

```
function redeemTicket(uint256 ticketNum, address referer) external {
    require(!address(msg.sender).isContract(), "ONLY_ALLOW_EOA");
    require(ticketNum >= 1 && ticketNum <= balanceOf(msg.sender),
"TICKET_NUM_INVALID");
    _burn(msg.sender,ticketNum);
    for (uint256 i = 0; i < ticketNum; i++) {
        _redeemSinglePrize(msg.sender, i, referer);
    }
}
```

- https://github.com/DODOEX/contractV2/blob/453e323af6/contracts/external/utils/Address.sol#L27

```
function isContract(address account) internal view returns (bool) {
    // This method relies on extcodesize, which returns 0 for contracts in
    // construction, since the code is only stored at the end of the
    // constructor execution.

    uint256 size;
    // solhint-disable-next-line no-inline-assembly
    assembly { size := extcodesize(account) }
    return size > 0;
}
```

Solution

It is recommended to add require(tx.origin == msg.sender); to check whether msg.sender is the contract address.

Status

Fixed; The issue has been fixed in commit: 5cda7ef2a3746e9025f4310e7bf830d5e1f4a0bf.

[N3] [Medium] Excessive authority issue

Category: Authority Control Vulnerability

Content

The owner can control the source of the seed of the random number. The seed of the random number will affect the value of the random number and affect the probability of redeeming the ticket.

- https://github.com/DODOEX/contractV2/blob/453e323af6/contracts/DODODrops/DODODropsV2/DODODrops.sol#L227

```
    function setRevealRn() external onlyOwner {
        require(_REVEAL_RN_ == 0, "ALREADY_SET");
        _REVEAL_RN_ = uint256(keccak256(abi.encodePacked(blockhash(block.number -
1))));
        emit SetReveal();
    }
```

The owner can change the value of `_RNG_`, which will affect the random number of the redemption ticket if it is not REVEAL_MODE.

- https://github.com/DODOEX/contractV2/blob/453e323af6/contracts/DODODrops/DODODropsV2/DODODrops.sol#L259

```
function updateRNG(address newRNG) external onlyOwner {
    require(newRNG != address(0));
    _RNG_ = newRNG;
    emit ChangeRNG(newRNG);
}
```

The owner can transfer `_REWARD_TOKEN_` to any address. The current design framework Owner address will be sent to the address of the Mine contract.

- https://github.com/DODOEX/contractV2/blob/c7202eeae7/contracts/DODOToken/DODOMineV3/RewardVault.sol#L38-L49

```
function reward(address to, uint256 amount) external onlyOwner {
    require(_REWARD_RESERVE_ >= amount, "VAULT_NOT_ENOUGH");
    _REWARD_RESERVE_ = _REWARD_RESERVE_.sub(amount);
    IERC20(_REWARD_TOKEN_).safeTransfer(to, amount);
}

function withdrawLeftOver(address to,uint256 amount) external onlyOwner {
    require(_REWARD_RESERVE_ >= amount, "VAULT_NOT_ENOUGH");
    _REWARD_RESERVE_ = _REWARD_RESERVE_.sub(amount);
    IERC20(_REWARD_TOKEN_).safeTransfer(to, amount);
}
```

The owner can mint tokens for any user and burn any user's tokens.

- https://github.com/DODOEX/contractV2/blob/c7202eeae7/contracts/external/ERC20/CustomERC20.sol#L123-L138

```
function mint(address user, uint256 value) external onlyOwner {
    require(isMintable, "NOT_MINTABEL_TOKEN");
    balances[user] = balances[user].add(value);
    totalSupply = totalSupply.add(value);
    emit Mint(user, value);
    emit Transfer(address(0), user, value);
}

function burn(address user, uint256 value) external onlyOwner {
    require(isMintable, "NOT_MINTABEL_TOKEN");
    balances[user] = balances[user].sub(value);
    totalSupply = totalSupply.sub(value);
    emit Burn(user, value);
    emit Transfer(user, address(0), value);
}
```

The owner can update the template contract. If an unaudited template contract is updated, this will affect the assets of the new user in the newly created contract.

- https://github.com/DODOEX/contractV2/blob/c7202eeae7/contracts/SmartRoute/proxies/DODOMineV3Proxy.sol#L128

```
function updateMineV2Template(address _newMineV3Template) external onlyOwner {
    _MINEV3_TEMPLATE_ = _newMineV3Template;
}
```

- https://github.com/DODOEX/contractV2/blob/7e629d0e58/contracts/Factory/ERC20V2Factory.sol

```
function updateStdTemplate(address newStdTemplate) external onlyOwner {
    _ERC20_TEMPLATE_ = newStdTemplate;
    emit ChangeStdTemplate(newStdTemplate);
}

function updateCustomTemplate(address newCustomTemplate) external onlyOwner {
    _CUSTOM_ERC20_TEMPLATE_ = newCustomTemplate;
    emit ChangeCustomTemplate(newCustomTemplate);
}
```

Solution

1. It is recommended to use future block hash or chainlink as a method to obtain random number seeds in the DODODrops contract.
2. It is recommended that no modification is allowed after confirming `_RNG_` in the DODODrops contract.
3. In the CustomERC20 contract, the Owner can mint and burn tokens for any user. It is recommended to delete the logic that the owner can manipulate other users' assets.
4. The owner in the ERC20V2Factory contract can change the template contract, if an unaudited template contract is updated, this will affect the assets of the new user in the newly created contract. It is recommended to set the Owner to a timelock contract or use governance to restrict it.

Status

Confirmed; The owner of the CustomERC20 contract can mint and burn tokens for any user, this issue has been fixed in commit: b0e91d2a092c6994a420b375a680a123af498e52 and commit: eed61b50d4df52d07717368b70d094a6d10f73c7.

[N4] [Suggestion] The DoS risk

Category: Others

Content

Use a for loop to traverse the array. If the number of loops is large, it will cause an out of gas. After communication and feedback, the project team will ensure that the number of rewardTokenInfos will not be too much.

- https://github.com/DODOEX/contractV2/blob/c7202eeae7/contracts/DODOToken/DODOMineV3/BaseMine.sol#L258

```
function _updateAllReward(address user) internal {
    uint256 len = rewardTokenInfos.length;
    for (uint256 i = 0; i < len; i++) {
        _updateReward(user, i);
    }
}
```

Solution

It is recommended that if the array has many elements, it can be called in batches to avoid the issue of out of gas.

Status

Confirmed

[N5] [Suggestion] Event log missing

Category: Others

Content

The owner can arbitrarily set an external contract address as a template contract. When a user creates a new contract, it will be created based on the template contract. After creation, the asset needs to be recharged to the new contract. There is no event record, which is unfavorable for review by community users.

- https://github.com/DODOEX/contractV2/blob/c7202eeae7/contracts/SmartRoute/proxies/DODOMineV3Proxy.sol#L128

```
function updateMineV2Template(address _newMineV3Template) external onlyOwner {
    _MINEV3_TEMPLATE_ = _newMineV3Template;
}
```

The owner can modify the configuration of the contract, but there is no event record, which is unfavorable for review by community users.

- https://github.com/DODOEX/contractV2/blob/c7202eeae7/contracts/Factory/Registries/DODOMineV3Registry.sol#L87-L101

```
function addAdminList (address contractAddr) external onlyOwner {
        isAdminListed[contractAddr] = true;
    }

function removeAdminList (address contractAddr) external onlyOwner {
    isAdminListed[contractAddr] = false;
}

function addSingleTokenList(address token) external onlyOwner {
    singleTokenList[token] = true;
}

function removeSingleTokenList(address token) external onlyOwner {
    singleTokenList[token] = false;
}
```

Solution

It is recommended to add event logs for recording to facilitate community review of the project.

Status

Confirmed

[N6] [Suggestion] Check enhancement of isLpToken

Category: Design Logic Audit

Content

Admin can add non-LPtoken assets but isLpToken is True, or belong to LPtoken assets but isLpToken is False Pool, which will affect the actual business logic. This part of the inspection is not implemented in the contract.

- https://github.com/DODOEX/contractV2/blob/c7202eeae7/contracts/Factory/Registries/DODOMineV3Registry.sol#L44

```
function addMineV3(
    address mine,
    bool isLpToken,
    address stakeToken
) override external {
    require(isAdminListed[msg.sender], "ACCESS_DENIED");
    _MINE_REGISTRY_[mine] = stakeToken;
    if(isLpToken) {
        _LP_REGISTRY_[stakeToken] = mine;
    }else {
        require(_SINGLE_REGISTRY_[stakeToken].length == 0 ||
singleTokenList[stakeToken], "ALREADY_EXSIT_POOL");
        _SINGLE_REGISTRY_[stakeToken].push(mine);
    }

    emit NewMineV3(mine, stakeToken, isLpToken);
}
```

Solution

It is recommended that Admin ensure that the input parameters are correct before adding Pool.

Status

Confirmed

[N7] [Suggestion] Security reminder on architecture design

Category: Others

Content

createStdERC20 and createMintableERC20 are open-ended calls. The user creates a contract using the createStdERC20 function to record the created information in `_USER_STD_REGISTRY_`, and then can get the information through getTokenByUser. Because it is an open call, it is not recommended to use the data obtained by getTokenByUser. As input for other businesses, after communication and feedback, the project party will not rely on the data obtained by getTokenByUser in the business logic of the project.

- https://github.com/DODOEX/contractV2/blob/c7202eeae7/contracts/Factory/ERC20V2Factory.sol#L72-L123

```
    function createStdERC20(
        uint256 totalSupply,
        string memory name,
        string memory symbol,
        uint256 decimals
    ) external returns (address newERC20) {
        newERC20 = ICloneFactory(_CLONE_FACTORY_).clone(_ERC20_TEMPLATE_);
        IStdERC20(newERC20).init(msg.sender, totalSupply, name, symbol, decimals);
        _USER_STD_REGISTRY_[msg.sender].push(newERC20);
        emit NewERC20(newERC20, msg.sender, 0);
    }

    function createCustomERC20(
        uint256 initSupply,
        string memory name,
        string memory symbol,
        uint256 decimals,
        uint256 tradeBurnRatio,
        uint256 tradeFeeRatio,
        address teamAccount,
        bool isMintable
    ) external returns (address newCustomERC20) {
        newCustomERC20 =
ICloneFactory(_CLONE_FACTORY_).clone(_CUSTOM_ERC20_TEMPLATE_);

        ICustomERC20(newCustomERC20).init(
            msg.sender,
            initSupply,
            name,
            symbol,
            decimals,
            tradeBurnRatio,
            tradeFeeRatio,
            teamAccount,
            isMintable
        );

        _USER_CUSTOM_REGISTRY_[msg.sender].push(newCustomERC20);
        if(isMintable)
            emit NewERC20(newCustomERC20, msg.sender, 2);
        else
            emit NewERC20(newCustomERC20, msg.sender, 1);
    }


    // ============ View ============
    function getTokenByUser(address user)
        external
        view
        returns (address[] memory stds,address[] memory customs)
    {
        return (_USER_STD_REGISTRY_[user], _USER_CUSTOM_REGISTRY_[user]);
    }
```

Solution

Because the data obtained by getTokenByUser is not credible, it is not recommended to use the data obtained by getTokenByUser as a dependency on other services.

Status

Confirmed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0x002107010001	SlowMist Security Team	2021.06.21 - 2021.06.30	Medium Risk

Summary conclusion: The SlowMist security team uses a manual and SlowMist team's analysis tool to audit the project, during the audit work we found a critical risk, two medium risks, four suggestions, and four suggestions were confirmed, and a medium risk vulnerability is fixed; A critical risk is incompletely fixed; The code was not deployed to the mainnet.

6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.