BEOSIN
Blockchain Security

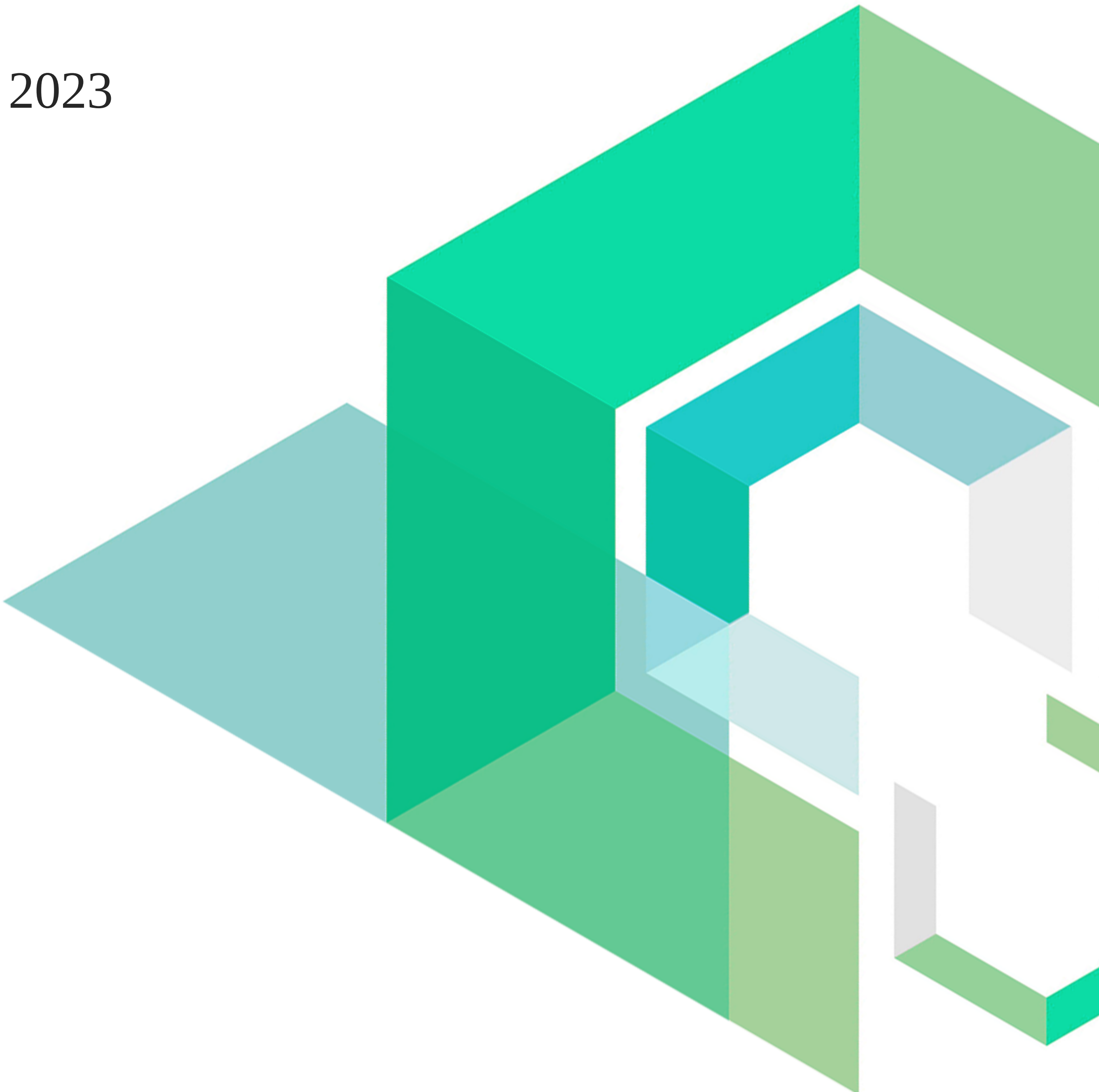
TimeFarm

Smart Contract Security Audit

V1.0

No. 202304111150

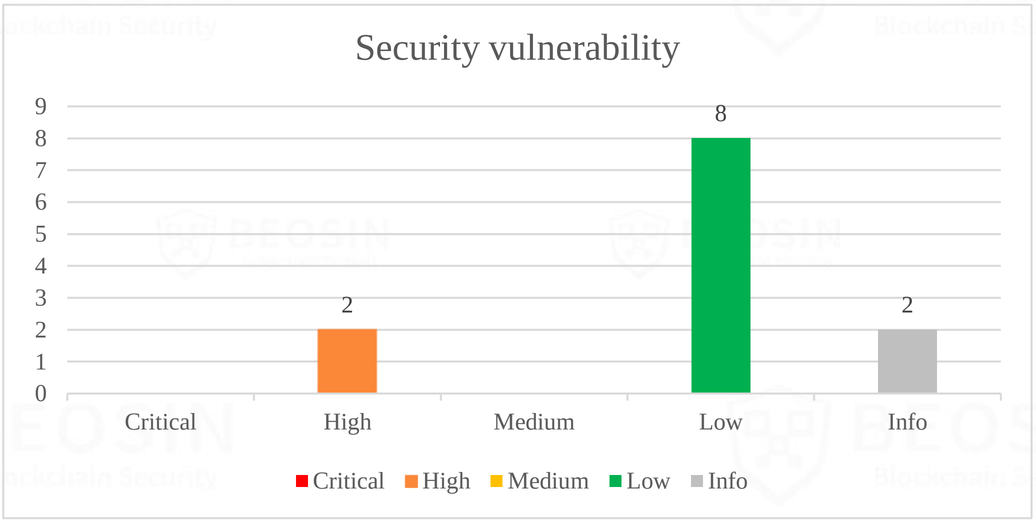
Apr 11th, 2023



Contents

Summary of Audit Results

After auditing, 2 High-risk, 8 Low-risks and 2 Info -risk were identified in the TimeFarm project. Specific audit details will be presented in the **Findings** section. Users should pay attention to the following aspects when interacting with this project:



*Notes:

- **Risk Description:**

Business overview

TimeFarm is a blockchain-based gaming platform. Functions such as token issuance and trading, team management, and blind box random rewards have been realized, allowing players to carry out economic activities and social interactions in the game, and gain fun and benefits.

1 Overview

1.1 Project Overview

Project Name	TimeFarm
Platform	FONChain

File Hash(SHA256)	01fe26e899985df29cb8758c5fb6bb8c70e6a3f70b9f558ac413713b0f22a422(V1)
	32f8f0c105b60754ae83647792dc3bec5570c8942e71639b7563bc8419f1eb71(V2)
	8af02135522938d99b051c3da76f18b64aa5a0de5eae41f82388405dd93bce6a(V3)
	56d9d7c05ac285caa0e176f84ee35383230d0b5b9135f4cb3e3c4f88b0dd042f(V4)
	709fce5f6f40c16c508c28f5e7403b3000dfa7d81c0c77f5e52780302ecd48ce(V5)

Audit scope	timefarm\contracts\FarmGame.sol timefarm\contracts\FarmGameBlindBox.sol timefarm\contracts\FarmGameTeam.sol timefarm\contracts\FarmNFT.sol timefarm\contracts\FarmToken.sol timefarm\contracts\Multicall.sol timefarm\contracts\interfaces\IFarmGame.sol timefarm\contracts\interfaces\IFarmNFT.sol timefarm\contracts\interfaces\IFarmTeam.sol timefarm\contracts \modules\FarmGameTool.sol timefarm\contracts\structs\FarmData.sol

1.2 Audit Overview

Audit work duration: Apr 06, 2023 – Apr 11, 2023

Audit methods: Formal Verification, Static Analysis, Typical Case Testing and Manual Review.

Audit team: Beosin Security Team.

2 Findings

Index	Risk description	Severity level	Status
TimeFarm-1	The random in *openBlindBoxItem* is controllable	**High**	Acknowledged
TimeFarm-2	The *takeVIP* did not check permissions	**High**	Fixed
TimeFarm-3	An team leader can remove themselves from a team	**Low**	Fixed
TimeFarm-4	The delete branch in *setBindBox* lacks an update to _blindBoxIds	**Low**	Fixed
TimeFarm-5	Unable to get the hash value of the current block	**Low**	Acknowledged
TimeFarm-6	The withdrawAmount calculation error	**Low**	Fixed
TimeFarm-7	The ecrecover function is missing a non-zero check for the signer	**Low**	Fixed
TimeFarm-8	Unsafe operation used	**Low**	Fixed
TimeFarm-9	Centralization risk	**Low**	Fixed
TimeFarm-10	The createTeamGoldAmount precision error	**Low**	Fixed
TimeFarm-11	Code redundancy	**info**	Fixed
TimeFarm-12	The key function add event	**info**	Acknowledged

Status Notes:

1. TimeFarm-1 is not fixed. The random number of opening the blind box is predictable. After the user obtains the signature data of the validator, the corresponding winning random number can be generated according to the number of prizes that have not yet been drawn. Then, the user can also wait until the number of rewards that has not yet been drawn becomes A number that is beneficial to yourself and then go to open blind box.

   The project party indicated that when the project is actually running, the number of rewards and the number of blind boxes are large, so the premise of waiting for the change of until the number of rewards to make a profit is that the attacker's random number is very close to the next prize; in addition, other random numbers in the contract in controllable business logic, the corresponding rewards are relatively low.

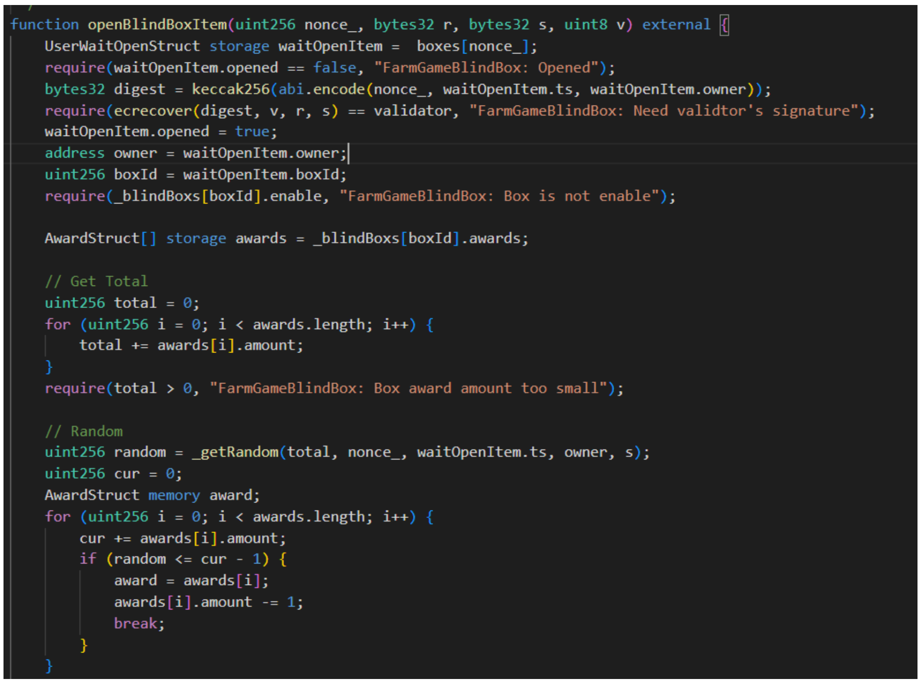
2. TimeFarm-5 is not fixed. The blockHash return value of the *tryBlockAndAggregate* function in the Multicall contract will always be 0.

   The project party stated that this contract is an auxiliary contract, and this interface will not be used and will not be processed.

Finding Details:

[TimeFarm-1] The random in *openBlindBoxItem* is controllable

Severity Level	High
Type	Business Security
Lines	FarmGameBlindBox.sol #L139-180
Description	This function is used to open the items in a blind box. If a user obtains parameters r, s, and v through an off-chain method, they can calculate the value of "random" ahead of time by observing the change in "total", since nonce_, waitOpenItem.ts, owner, and s are already known. This allows the user to obtain the award at an appropriate time.



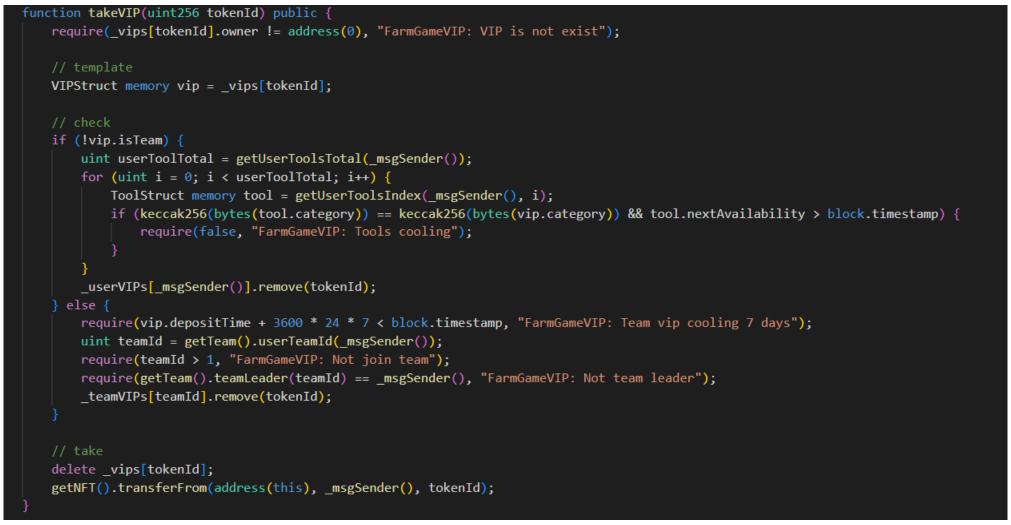
```
function openBlindBoxItem(uint256 nonce_, bytes32 r, bytes32 s, uint8 v) external {
    UserWaitOpenStruct storage waitOpenItem =  boxes[nonce_];
    require(waitOpenItem.opened == false, "FarmGameBlindBox: Opened");
    bytes32 digest = keccak256(abi.encode(nonce_, waitOpenItem.ts, waitOpenItem.owner));
    require(ecrecover(digest, v, r, s) == validator, "FarmGameBlindBox: Need validtor's signature");
    waitOpenItem.opened = true;
    address owner = waitOpenItem.owner;
    uint256 boxId = waitOpenItem.boxId;
    require(_blindBoxs[boxId].enable, "FarmGameBlindBox: Box is not enable");

    AwardStruct[] storage awards = _blindBoxs[boxId].awards;

    // Get Total
    uint256 total = 0;
    for (uint256 i = 0; i < awards.length; i++) {
        total += awards[i].amount;
    }
    require(total > 0, "FarmGameBlindBox: Box award amount too small");

    // Random
    uint256 random = _getRandom(total, nonce_, waitOpenItem.ts, owner, s);
    uint256 cur = 0;
    AwardStruct memory award;
    for (uint256 i = 0; i < awards.length; i++) {
        cur += awards[i].amount;
        if (random <= cur - 1) {
            award = awards[i];
            awards[i].amount -= 1;
            break;
        }
    }
```

Figure 1 Source code of *openBlindBoxItem* function

Recommendations	It is recommended to using Chainlink VRF tool to provide random numbers.
Status	Acknowledged. The controllability of random numbers has already reduced the rewards.

[TimeFarm-2] The *takeVIP* did not check permissions

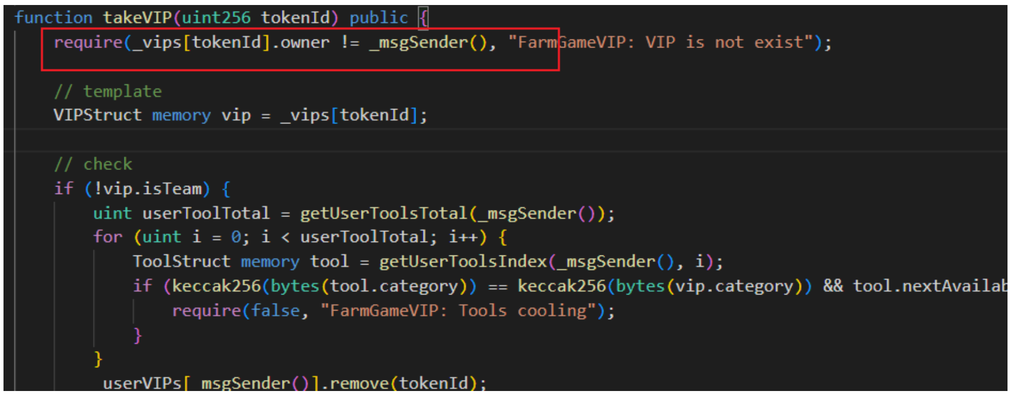
Severity Level	High
Type	Business Security
Lines	FarmGameTool.sol #L270-297
Description	The contract does not check the relationship between msg.sender and tokenid. When the attacker uses remove to remove, if the corresponding element does not exist, the contract will not report an error, but return false, but this contract does not check the return value, resulting in allowing The attacker steals the NFT in the contract.



```
function takeVIP(uint256 tokenId) public {
    require(_vips[tokenId].owner != address(0), "FarmGameVIP: VIP is not exist");

    // template
    VIPStruct memory vip = _vips[tokenId];

    // check
    if (!vip.isTeam) {
        uint userToolTotal = getUserToolsTotal(_msgSender());
        for (uint i = 0; i < userToolTotal; i++) {
            ToolStruct memory tool = getUserToolsIndex(_msgSender(), i);
            if (keccak256(bytes(tool.category)) == keccak256(bytes(vip.category)) && tool.nextAvailability > block.timestamp) {
                require(false, "FarmGameVIP: Tools cooling");
            }
        }
        _userVIPs[_msgSender()].remove(tokenId);
    } else {
        require(vip.depositTime + 3600 * 24 * 7 < block.timestamp, "FarmGameVIP: Team vip cooling 7 days");
        uint teamId = getTeam().userTeamId(_msgSender());
        require(teamId > 1, "FarmGameVIP: Not join team");
        require(getTeam().teamLeader(teamId) == _msgSender(), "FarmGameVIP: Not team leader");
        _teamVIPs[teamId].remove(tokenId);
    }

    // take
    delete _vips[tokenId];
    getNFT().transferFrom(address(this), _msgSender(), tokenId);
}
```

Figure 2 Source code of *takeVIP* function

Recommendations	It is recommended to check the relationship between msg.sender and tokenid.
Status	Fixed.

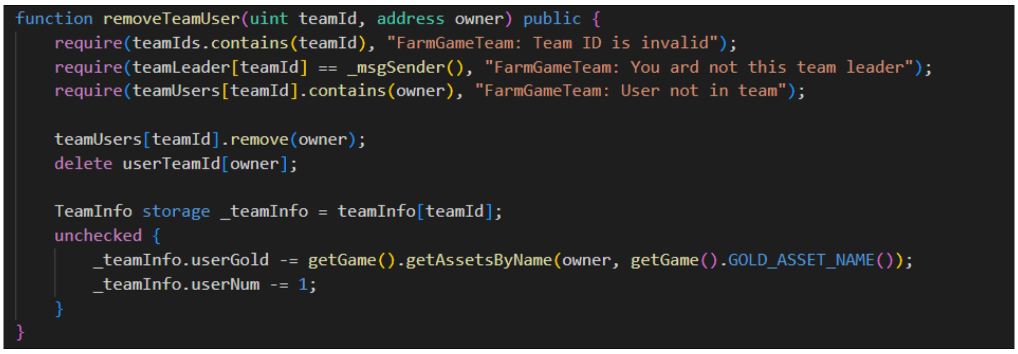


```
function takeVIP(uint256 tokenId) public {
    require(_vips[tokenId].owner != _msgSender(), "FarmGameVIP: VIP is not exist");

    // template
    VIPStruct memory vip = _vips[tokenId];

    // check
    if (!vip.isTeam) {
        uint userToolTotal = getUserToolsTotal(_msgSender());
        for (uint i = 0; i < userToolTotal; i++) {
            ToolStruct memory tool = getUserToolsIndex(_msgSender(), i);
            if (keccak256(bytes(tool.category)) == keccak256(bytes(vip.category)) && tool.nextAvailab
                require(false, "FarmGameVIP: Tools cooling");
            }
        }
        _userVIPs[_msgSender()].remove(tokenId);
```

Figure 3 Source code of *takeVIP* function(fixed)

[TimeFarm-3] An team leader can remove themselves from a team

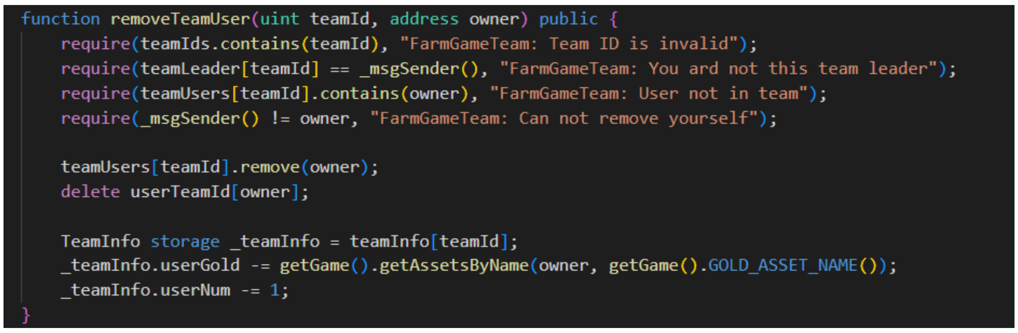
Severity Level	Low
Type	Business Security
Lines	FarmGameTeam.sol#L192-204
Description	When a team leader enters the address for deleting an team leader, they can remove the team leader from the team, bypassing the restriction on deleting team leader in the *exitTeam* function.



```
function removeTeamUser(uint teamId, address owner) public {
    require(teamIds.contains(teamId), "FarmGameTeam: Team ID is invalid");
    require(teamLeader[teamId] == _msgSender(), "FarmGameTeam: You ard not this team leader");
    require(teamUsers[teamId].contains(owner), "FarmGameTeam: User not in team");

    teamUsers[teamId].remove(owner);
    delete userTeamId[owner];

    TeamInfo storage _teamInfo = teamInfo[teamId];
    unchecked {
        _teamInfo.userGold -= getGame().getAssetsByName(owner, getGame().GOLD_ASSET_NAME());
        _teamInfo.userNum -= 1;
    }
}
```

Figure 4 Source code of *removeTeamUser* function

Recommendations	It is recommended to determine whether the owner address is an team leader address.
Status	Fixed.



```
function removeTeamUser(uint teamId, address owner) public {
    require(teamIds.contains(teamId), "FarmGameTeam: Team ID is invalid");
    require(teamLeader[teamId] == _msgSender(), "FarmGameTeam: You ard not this team leader");
    require(teamUsers[teamId].contains(owner), "FarmGameTeam: User not in team");
    require(_msgSender() != owner, "FarmGameTeam: Can not remove yourself");

    teamUsers[teamId].remove(owner);
    delete userTeamId[owner];

    TeamInfo storage _teamInfo = teamInfo[teamId];
    _teamInfo.userGold -= getGame().getAssetsByName(owner, getGame().GOLD_ASSET_NAME());
    _teamInfo.userNum -= 1;
}
```

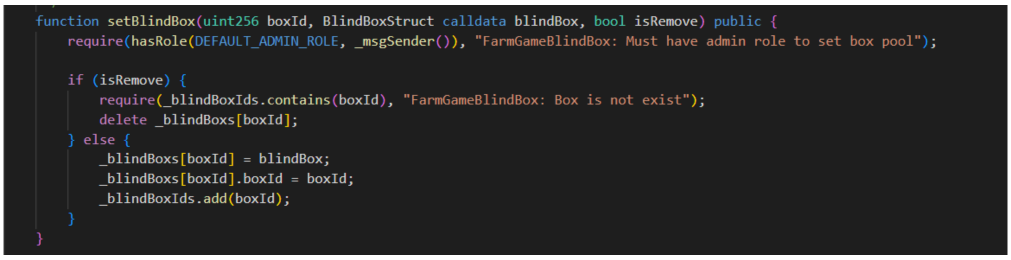
Figure 5 Source code of *removeTeamUser* function(fixed)

[TimeFarm-4] The delete branch in *setBindBox* lacks an update to _blindBoxIds

Severity Level	Low
Type	Business Security
Lines	FarmGameBlindBox.sol #L87-98

Description

When isRemove is true, boxId will be removed from _blindBoxs, but the case of _blindBoxIds is not considered, and boxId is not removed from _blindBoxIds.



```
    function setBlindBox(uint256 boxId, BlindBoxStruct calldata blindBox, bool isRemove) public {
        require(hasRole(DEFAULT_ADMIN_ROLE, _msgSender()), "FarmGameBlindBox: Must have admin role to set box pool");

        if (isRemove) {
            require(_blindBoxIds.contains(boxId), "FarmGameBlindBox: Box is not exist");
            delete _blindBoxs[boxId];
        } else {
            _blindBoxs[boxId] = blindBox;
            _blindBoxs[boxId].boxId = boxId;
            _blindBoxIds.add(boxId);
        }
    }
```

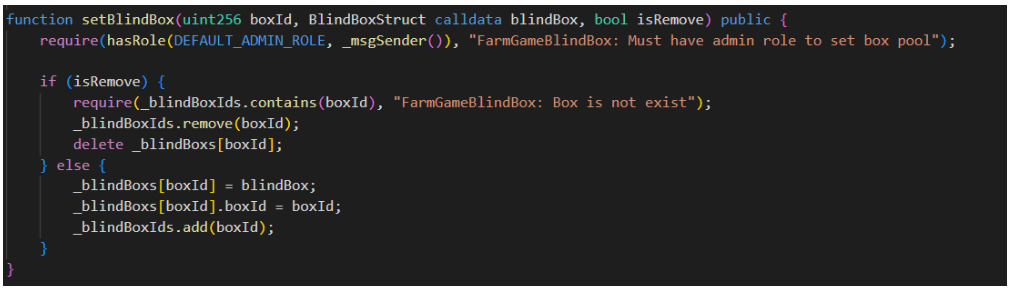
Figure 6 Source code of *setBlindBox* function

Recommendations

It is recommended to in the branch where isRemove is true, add _blindBoxIds.remove(boxId).

Status

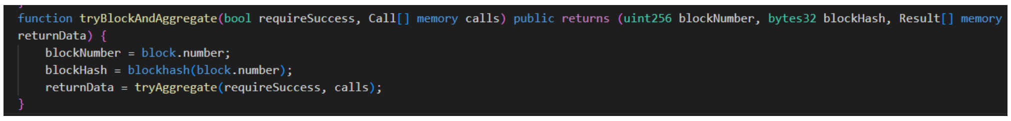
Fixed.



```
function setBlindBox(uint256 boxId, BlindBoxStruct calldata blindBox, bool isRemove) public {
    require(hasRole(DEFAULT_ADMIN_ROLE, _msgSender()), "FarmGameBlindBox: Must have admin role to set box pool");

    if (isRemove) {
        require(_blindBoxIds.contains(boxId), "FarmGameBlindBox: Box is not exist");
        _blindBoxIds.remove(boxId);
        delete _blindBoxs[boxId];
    } else {
        _blindBoxs[boxId] = blindBox;
        _blindBoxs[boxId].boxId = boxId;
        _blindBoxIds.add(boxId);
    }
}
```

Figure 7 Source code of *setBlindBox* function(fixed)

[TimeFarm-5] Unable to get the hash value of the current block

Severity Level	**Low**
Type	Business Security
Lines	Multicall.sol #L69-73
Description	Use the blockhash (blockNumber) function to obtain the hash value of the specified block number, but *blockhash* cannot query the hash value of the current block because the hash value of the current block is not yet available.  `function tryBlockAndAggregate(bool requireSuccess, Call[] memory calls) public returns (uint256 blockNumber, bytes32 blockHash, Result[] memory returnData) { blockNumber = block.number; blockHash = blockhash(block.number); returnData = tryAggregate(requireSuccess, calls); }` Figure 8 Source code of *tryBlockAndAggregate* function
Recommendations	It is recommended to use blockhash(block.number - 1).
Status	Acknowledged.

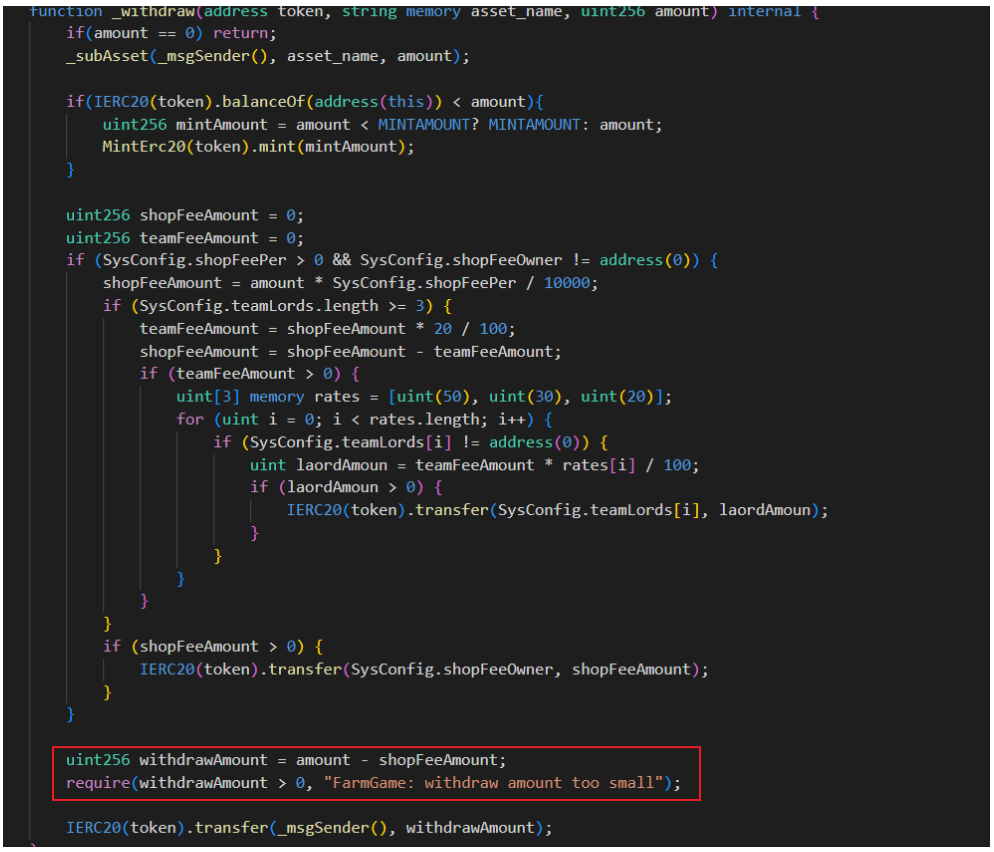
```
function tryBlockAndAggregate(bool requireSuccess, Call[] memory calls) public returns (uint256 blockNumber, bytes32 blockHash, Result[] memory
returnData) {
    blockNumber = block.number;
    blockHash = blockhash(block.number);
    returnData = tryAggregate(requireSuccess, calls);
}
```

[TimeFarm-6] The withdrawAmount calculation error

Severity Level	**Low**
Type	Business Security
Lines	FarmGame.sol #L296

Description

After the shopfee and teamfee are calculated, the final withdraw amount should be subtracted from teamFeeAmount instead of keeping it as it is. Failure to modify will result in an excessively high amount ultimately sent to the caller and may result in unintended loss of funds.



```
function _withdraw(address token, string memory asset_name, uint256 amount) internal {
    if(amount == 0) return;
    _subAsset(_msgSender(), asset_name, amount);

    if(IERC20(token).balanceOf(address(this)) < amount){
        uint256 mintAmount = amount < MINTAMOUNT? MINTAMOUNT: amount;
        MintErc20(token).mint(mintAmount);
    }

    uint256 shopFeeAmount = 0;
    uint256 teamFeeAmount = 0;
    if (SysConfig.shopFeePer > 0 && SysConfig.shopFeeOwner != address(0)) {
        shopFeeAmount = amount * SysConfig.shopFeePer / 10000;
        if (SysConfig.teamLords.length >= 3) {
            teamFeeAmount = shopFeeAmount * 20 / 100;
            shopFeeAmount = shopFeeAmount - teamFeeAmount;
            if (teamFeeAmount > 0) {
                uint[3] memory rates = [uint(50), uint(30), uint(20)];
                for (uint i = 0; i < rates.length; i++) {
                    if (SysConfig.teamLords[i] != address(0)) {
                        uint laordAmoun = teamFeeAmount * rates[i] / 100;
                        if (laordAmoun > 0) {
                            IERC20(token).transfer(SysConfig.teamLords[i], laordAmoun);
                        }
                    }
                }
            }
        }
        if (shopFeeAmount > 0) {
            IERC20(token).transfer(SysConfig.shopFeeOwner, shopFeeAmount);
        }
    }

    uint256 withdrawAmount = amount - shopFeeAmount;
    require(withdrawAmount > 0, "FarmGame: withdraw amount too small");

    IERC20(token).transfer(_msgSender(), withdrawAmount);
```

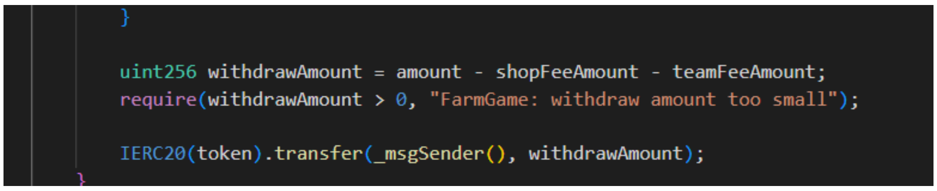
Figure 9 Source code of _*withdraw* function

Recommendations

It is recommended to modify it to withdrawAmount = amount - shopFeeAmount – teamFeeAmount.

Status

Fixed.

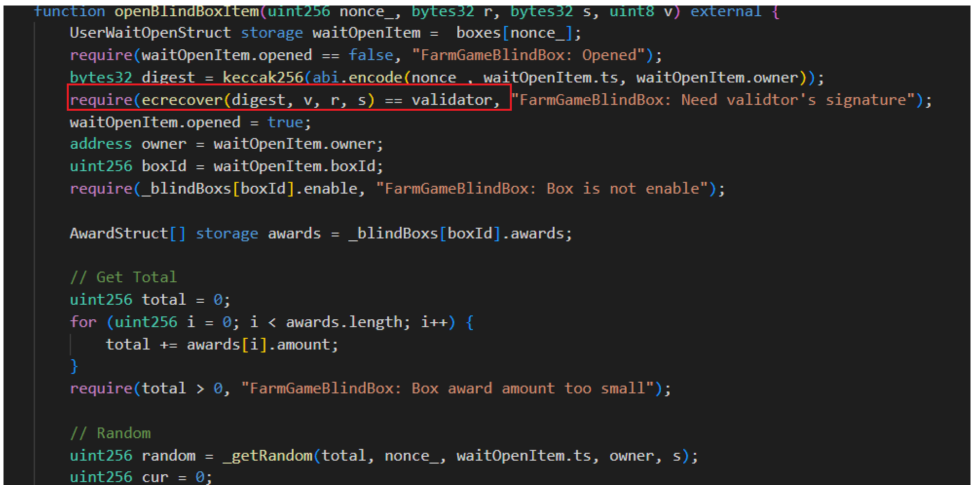


```
        }

        uint256 withdrawAmount = amount - shopFeeAmount - teamFeeAmount;
        require(withdrawAmount > 0, "FarmGame: withdraw amount too small");

        IERC20(token).transfer(_msgSender(), withdrawAmount);
    }
```

Figure 10 Source code of _*withdraw* function(fixed)

[TimeFarm-7] The ecrecover function is missing a non-zero check for the signer

Severity Level	**Low**
Type	Business Security
Lines	FarmGameBlindBox.sol #L87-98
Description	Check validator is not 0 address There is no zero address check for the validator variable. Therefore, a check should be added in this function to ensure that the validator is not a 0 address. If the validator is address 0, the execution of the function should be prohibited.



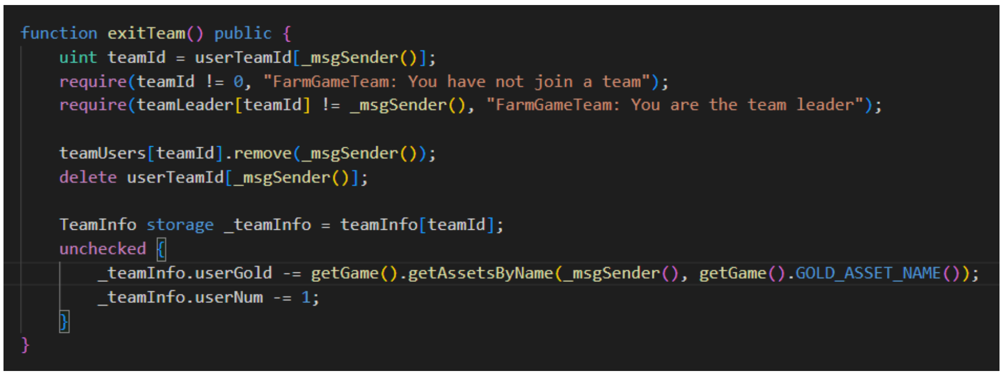
```
function openBlindBoxItem(uint256 nonce_, bytes32 r, bytes32 s, uint8 v) external {
    UserWaitOpenStruct storage waitOpenItem =  boxes[nonce_];
    require(waitOpenItem.opened == false, "FarmGameBlindBox: Opened");
    bytes32 digest = keccak256(abi.encode(nonce_, waitOpenItem.ts, waitOpenItem.owner));
    require(ecrecover(digest, v, r, s) == validator, "FarmGameBlindBox: Need validtor's signature");
    waitOpenItem.opened = true;
    address owner = waitOpenItem.owner;
    uint256 boxId = waitOpenItem.boxId;
    require(_blindBoxs[boxId].enable, "FarmGameBlindBox: Box is not enable");

    AwardStruct[] storage awards = _blindBoxs[boxId].awards;

    // Get Total
    uint256 total = 0;
    for (uint256 i = 0; i < awards.length; i++) {
        total += awards[i].amount;
    }
    require(total > 0, "FarmGameBlindBox: Box award amount too small");

    // Random
    uint256 random = _getRandom(total, nonce_, waitOpenItem.ts, owner, s);
    uint256 cur = 0;
```

Figure 11 Source code of *openBlindBoxItem* function

Recommendations	It is recommended to add a check to ensure that the validator is not a 0 address.
Status	Fixed.

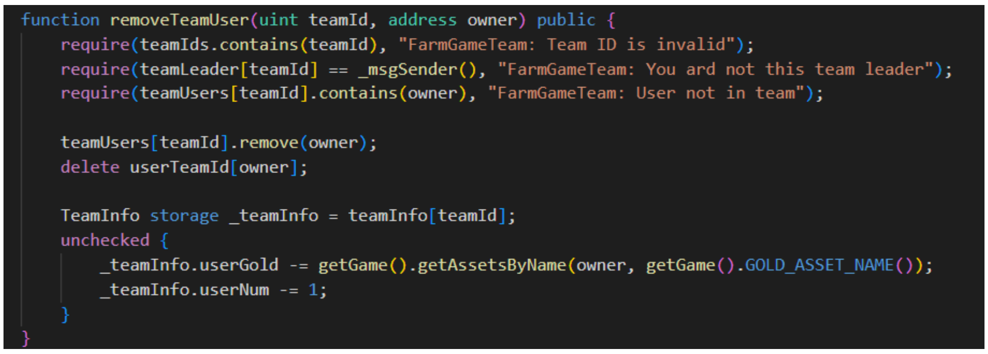
[TimeFarm-8] Unsafe operation used

Severity Level	Low
Type	Business Security
Lines	FarmGameTeam.sol #L200-L203,#L140-L143
Description	In the project contract, unchecked is directly used in many places and subtraction is performed. This is an unsafe approach. For example, when the assets of the exiting user are greater than the assets of the team as a whole, the value of userGold will overflow.



```
function exitTeam() public {
    uint teamId = userTeamId[_msgSender()];
    require(teamId != 0, "FarmGameTeam: You have not join a team");
    require(teamLeader[teamId] != _msgSender(), "FarmGameTeam: You are the team leader");

    teamUsers[teamId].remove(_msgSender());
    delete userTeamId[_msgSender()];

    TeamInfo storage _teamInfo = teamInfo[teamId];
    unchecked {
        _teamInfo.userGold -= getGame().getAssetsByName(_msgSender(), getGame().GOLD_ASSET_NAME());
        _teamInfo.userNum -= 1;
    }
}
```

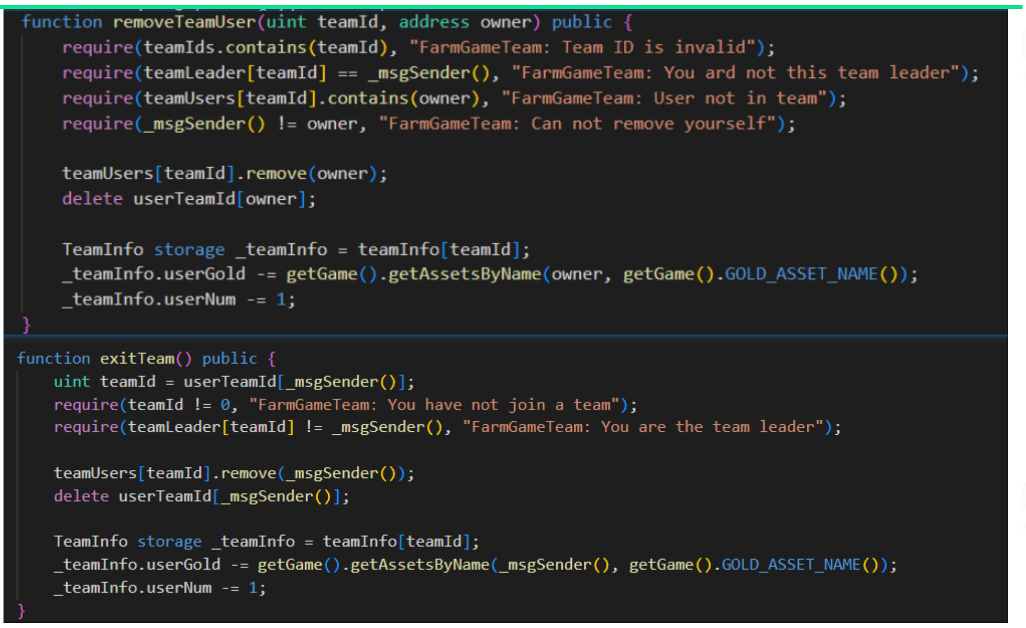
Figure 12 Source code of *exitTeam* function



```
function removeTeamUser(uint teamId, address owner) public {
    require(teamIds.contains(teamId), "FarmGameTeam: Team ID is invalid");
    require(teamLeader[teamId] == _msgSender(), "FarmGameTeam: You ard not this team leader");
    require(teamUsers[teamId].contains(owner), "FarmGameTeam: User not in team");

    teamUsers[teamId].remove(owner);
    delete userTeamId[owner];

    TeamInfo storage _teamInfo = teamInfo[teamId];
    unchecked {
        _teamInfo.userGold -= getGame().getAssetsByName(owner, getGame().GOLD_ASSET_NAME());
        _teamInfo.userNum -= 1;
    }
}
```

Figure 13 Source code of *removeTeamUser* function

Recommendations	It is recommended to check the size of the value before performing the subtraction budget.
Status	Fixed.


```
function removeTeamUser(uint teamId, address owner) public {
    require(teamIds.contains(teamId), "FarmGameTeam: Team ID is invalid");
    require(teamLeader[teamId] == _msgSender(), "FarmGameTeam: You ard not this team leader");
    require(teamUsers[teamId].contains(owner), "FarmGameTeam: User not in team");
    require(_msgSender() != owner, "FarmGameTeam: Can not remove yourself");

    teamUsers[teamId].remove(owner);
    delete userTeamId[owner];

    TeamInfo storage _teamInfo = teamInfo[teamId];
    _teamInfo.userGold -= getGame().getAssetsByName(owner, getGame().GOLD_ASSET_NAME());
    _teamInfo.userNum -= 1;
}
```

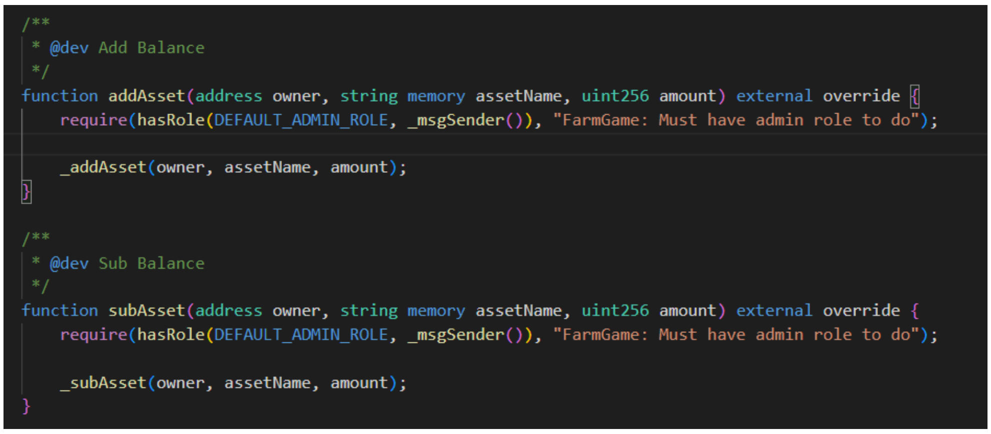
```
function exitTeam() public {
    uint teamId = userTeamId[_msgSender()];
    require(teamId != 0, "FarmGameTeam: You have not join a team");
    require(teamLeader[teamId] != _msgSender(), "FarmGameTeam: You are the team leader");

    teamUsers[teamId].remove(_msgSender());
    delete userTeamId[_msgSender()];

    TeamInfo storage _teamInfo = teamInfo[teamId];
    _teamInfo.userGold -= getGame().getAssetsByName(_msgSender(), getGame().GOLD_ASSET_NAME());
    _teamInfo.userNum -= 1;
}
```

Figure 14 Source code of *removeTeamUser* and *exitTeam* function(fixed)

[TimeFarm-9] Centralization risk

Severity Level	Low
Type	Business Security
Lines	FarmGame.sol #L324-L337

Description Administrators can modify any asset for any account.

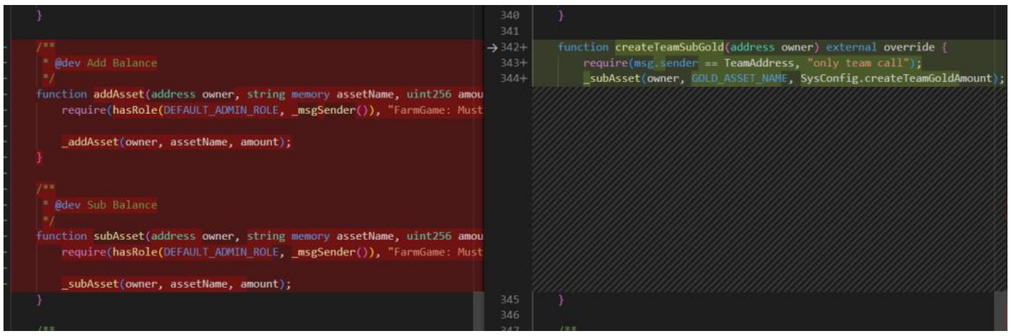


```
/**
 * @dev Add Balance
 */
function addAsset(address owner, string memory assetName, uint256 amount) external override {
    require(hasRole(DEFAULT_ADMIN_ROLE, _msgSender()), "FarmGame: Must have admin role to do");

    _addAsset(owner, assetName, amount);
}

/**
 * @dev Sub Balance
 */
function subAsset(address owner, string memory assetName, uint256 amount) external override {
    require(hasRole(DEFAULT_ADMIN_ROLE, _msgSender()), "FarmGame: Must have admin role to do");

    _subAsset(owner, assetName, amount);
}
```

Figure 15 Source code of *addAsset* and *subAsset* function

Recommendations It is recommended to delete the corresponding function.

Status Fixed. Add a new *createTeamSubGold* function for unified operation.



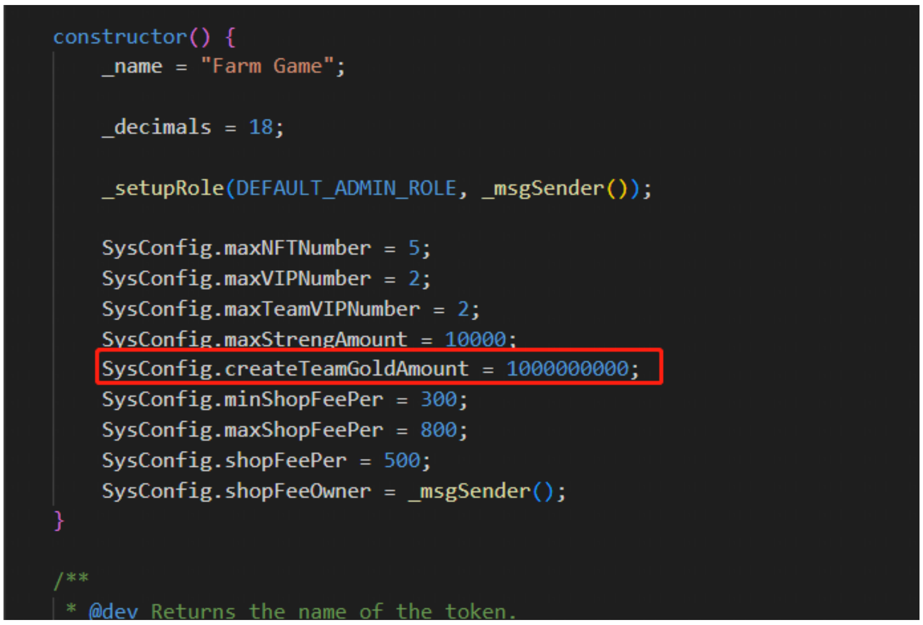
```
}

/**
 * @dev Add Balance
 */
function addAsset(address owner, string memory assetName, uint256 amou
    require(hasRole(DEFAULT_ADMIN_ROLE, _msgSender()), "FarmGame: Must

    _addAsset(owner, assetName, amount);
}

/**
 * @dev Sub Balance
 */
function subAsset(address owner, string memory assetName, uint256 amou
    require(hasRole(DEFAULT_ADMIN_ROLE, _msgSender()), "FarmGame: Must

    _subAsset(owner, assetName, amount);
}
```

```
}

function createTeamSubGold(address owner) external override {
    require(msg.sender == TeamAddress, "only team call");
    _subAsset(owner, GOLD_ASSET_NAME, SysConfig.createTeamGoldAmount);
}
```

Figure 16 Source code of *createTeamSubGold* function(fixed)

[TimeFarm-10] The createTeamGoldAmount precision error

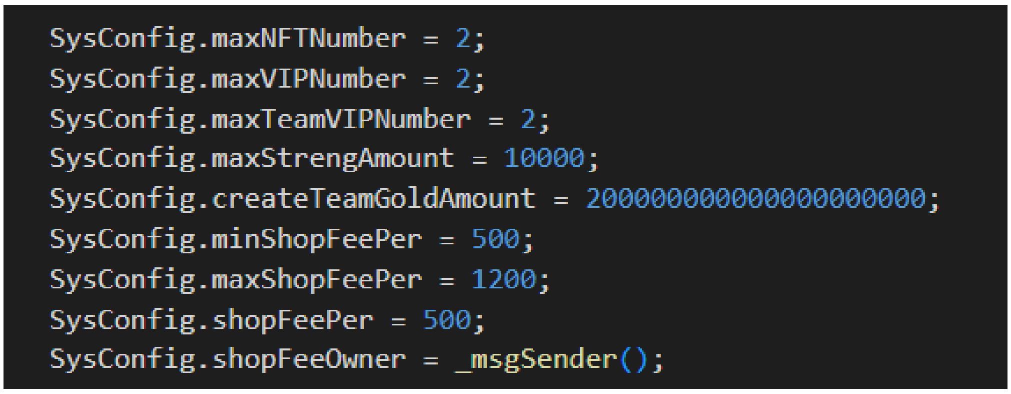
Severity Level	Low
Type	Business Security
Lines	FarmGame.sol #L55
Description	The createTeamGoldAmount precision is 18 digits, the default value in the *constructor* is too small.



```
constructor() {
    _name = "Farm Game";

    _decimals = 18;

    _setupRole(DEFAULT_ADMIN_ROLE, _msgSender());

    SysConfig.maxNFTNumber = 5;
    SysConfig.maxVIPNumber = 2;
    SysConfig.maxTeamVIPNumber = 2;
    SysConfig.maxStrengAmount = 10000;
    SysConfig.createTeamGoldAmount = 1000000000;
    SysConfig.minShopFeePer = 300;
    SysConfig.maxShopFeePer = 800;
    SysConfig.shopFeePer = 500;
    SysConfig.shopFeeOwner = _msgSender();
}

/**
 * @dev Returns the name of the token.
```

Figure 17 Source code of *constructor* function

Recommendations	It is recommended to be consistent with the accuracy of goldtoken.
Status	Fixed.

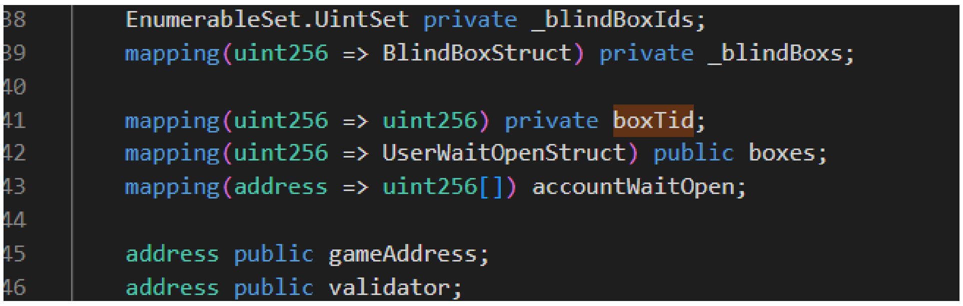


```
SysConfig.maxNFTNumber = 2;
SysConfig.maxVIPNumber = 2;
SysConfig.maxTeamVIPNumber = 2;
SysConfig.maxStrengAmount = 10000;
SysConfig.createTeamGoldAmount = 20000000000000000000;
SysConfig.minShopFeePer = 500;
SysConfig.maxShopFeePer = 1200;
SysConfig.shopFeePer = 500;
SysConfig.shopFeeOwner = _msgSender();
```

Figure 18 Source code of *constructor* function(fixed)

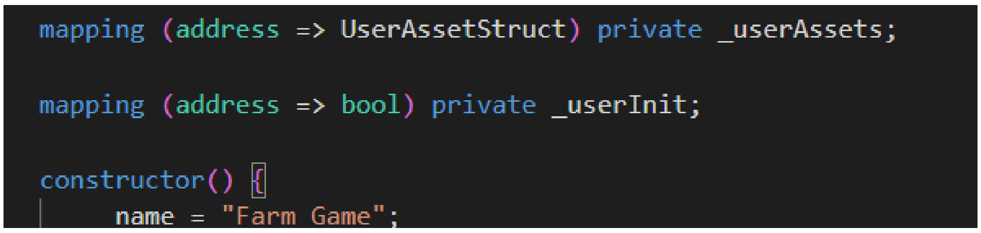
[TimeFarm-11] Code redundancy

Severity Level	Info
Type	Coding Conventions
Lines	FarmGameBlindBox.sol #L41,FarmGame#L41
Description	The boxTid in the FarmGameBlindBox contract is not used, and the _userInit in the FarmGame contract is not used.



```
    EnumerableSet.UintSet private _blindBoxIds;
    mapping(uint256 => BlindBoxStruct) private _blindBoxs;

    mapping(uint256 => uint256) private boxTid;
    mapping(uint256 => UserWaitOpenStruct) public boxes;
    mapping(address => uint256[]) accountWaitOpen;

    address public gameAddress;
    address public validator;
```

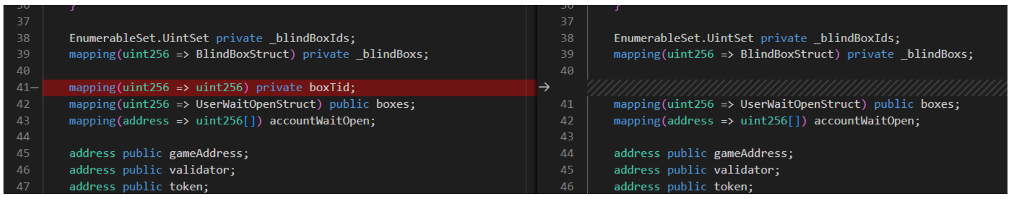
Figure 19 Source code of boxTid



```
mapping (address => UserAssetStruct) private _userAssets;

mapping (address => bool) private _userInit;

constructor() {
    name = "Farm Game";
```

Figure 20 Source code of _userInit

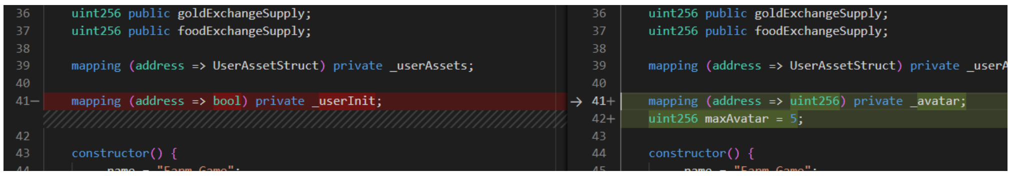
Recommendations	It is recommended to delete the code.
Status	Fixed.



```
    EnumerableSet.UintSet private _blindBoxIds;
    mapping(uint256 => BlindBoxStruct) private _blindBoxs;

    mapping(uint256 => uint256) private boxTid;
    mapping(uint256 => UserWaitOpenStruct) public boxes;
    mapping(address => uint256[]) accountWaitOpen;

    address public gameAddress;
    address public validator;
    address public token;
```

```
    EnumerableSet.UintSet private _blindBoxIds;
    mapping(uint256 => BlindBoxStruct) private _blindBoxs;

    mapping(uint256 => UserWaitOpenStruct) public boxes;
    mapping(address => uint256[]) accountWaitOpen;

    address public gameAddress;
    address public validator;
    address public token;
```

Figure 21 Source code after fix



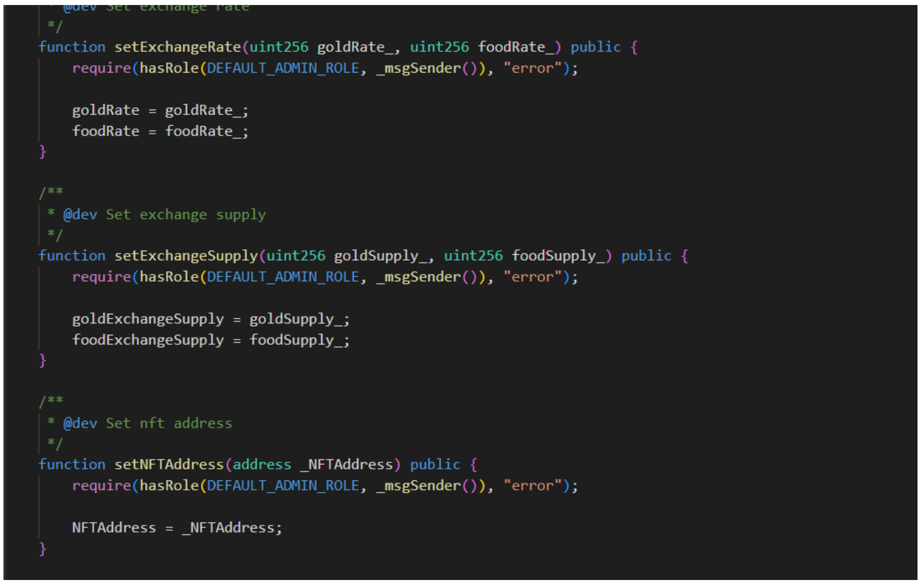
```
    uint256 public goldExchangeSupply;
    uint256 public foodExchangeSupply;

    mapping (address => UserAssetStruct) private _userAssets;

    mapping (address => bool) private _userInit;

    constructor() {
        name = "Farm Game";
```

```
    uint256 public goldExchangeSupply;
    uint256 public foodExchangeSupply;

    mapping (address => UserAssetStruct) private _userA

    mapping (address => uint256) private _avatar;
    uint256 maxAvatar = 5;

    constructor() {
        name = "Farm Game";
```

Figure 22 Source code after fix

[TimeFarm-12] The key function add event

Severity Level	Info
Type	Coding Conventions
Lines	FarmGameTeam.sol,FarmGame.sol,FarmGameBlindBox.sol
Description	In Ethereum smart contracts, when important state variables are modified, we usually want to know that this change has occurred. To do this, an event can be added to log the modification of the state variable.



```
    */
function setExchangeRate(uint256 goldRate_, uint256 foodRate_) public {
    require(hasRole(DEFAULT_ADMIN_ROLE, _msgSender()), "error");

    goldRate = goldRate_;
    foodRate = foodRate_;
}

/**
 * @dev Set exchange supply
 */
function setExchangeSupply(uint256 goldSupply_, uint256 foodSupply_) public {
    require(hasRole(DEFAULT_ADMIN_ROLE, _msgSender()), "error");

    goldExchangeSupply = goldSupply_;
    foodExchangeSupply = foodSupply_;
}

/**
 * @dev Set nft address
 */
function setNFTAddress(address _NFTAddress) public {
    require(hasRole(DEFAULT_ADMIN_ROLE, _msgSender()), "error");

    NFTAddress = _NFTAddress;
}
```

Figure 23 Source code of key function

Recommendations	It is recommended to add corresponding events for key operations.
Status	Acknowledged.

3 Appendix

3.1 Vulnerability Assessment Metrics and Status in Smart Contracts

3.1.1 Metrics

In order to objectively assess the severity level of vulnerabilities in blockchain systems, this report provides detailed assessment metrics for security vulnerabilities in smart contracts with reference to CVSS 3.1 (Common Vulnerability Scoring System Ver 3.1).

According to the severity level of vulnerability, the vulnerabilities are classified into four levels: "critical", "high", "medium" and "low". It mainly relies on the degree of impact and likelihood of exploitation of the vulnerability, supplemented by other comprehensive factors to determine of the severity level.

Likelihood \ Impact	Severe	High	Medium	Low
Probable	Critical	High	Medium	Low
Possible	High	High	Medium	Low
Unlikely	Medium	Medium	Low	Info
Rare	Low	Low	Info	Info

3.1.2 Degree of impact

- **Severe**

Severe impact generally refers to the vulnerability can have a serious impact on the confidentiality, integrity, availability of smart contracts or their economic model, which can cause substantial economic losses to the contract business system, large-scale data disruption, loss of authority management, failure of key functions, loss of credibility, or indirectly affect the operation of other smart contracts associated with it and cause substantial losses, as well as other severe and mostly irreversible harm.

- **High**

High impact generally refers to the vulnerability can have a relatively serious impact on the confidentiality, integrity, availability of the smart contract or its economic model, which can cause a greater economic loss, local functional unavailability, loss of credibility and other impact to the contract business system.

- **Medium**

Medium impact generally refers to the vulnerability can have a relatively minor impact on the confidentiality, integrity, availability of the smart contract or its economic model, which can cause a small amount of economic loss to the contract business system, individual business unavailability and other impact.

- **Low**

Low impact generally refers to the vulnerability can have a minor impact on the smart contract, which can pose certain security threat to the contract business system and needs to be improved.

3.1.4 Likelihood of Exploitation

- **Probable**

Probable likelihood generally means that the cost required to exploit the vulnerability is low, with no special exploitation threshold, and the vulnerability can be triggered consistently.

- **Possible**

Possible likelihood generally means that exploiting such vulnerability requires a certain cost, or there are certain conditions for exploitation, and the vulnerability is not easily and consistently triggered.

- **Unlikely**

Unlikely likelihood generally means that the vulnerability requires a high cost, or the exploitation conditions are very demanding and the vulnerability is highly difficult to trigger.

- **Rare**

Rare likelihood generally means that the vulnerability requires an extremely high cost or the conditions for exploitation are extremely difficult to achieve.

3.1.5 Fix Results Status

Status	**Description**
Fixed	The project party fully fixes a vulnerability.
Partially Fixed	The project party did not fully fix the issue, but only mitigated the issue.
Acknowledged	The project party confirms and chooses to ignore the issue.

3.2 Audit Categories

No.	Categories	Subitems
1	Coding Conventions	Redundant Code
		require/assert Usage
		Cycles Consumption
2	General Vulnerability	Integer Overflow/Underflow
		Reentrancy
		Pseudo-random Number Generator (PRNG)
		Transaction-Ordering Dependence
		DoS (Denial of Service)
		Function Call Permissions
		Returned Value Security
		Rollback Risk
		Replay Attack
		Overriding Variables
		Call Canister controllable
		Canister upgrade risk
		Third-party Protocol Interface Consistency
3	Business Security	Business Logics
		Business Implementations
		Manipulable Token Price
		Centralized Asset Control
		Asset Tradability
		Arbitrage Attack

Beosin classified the security issues of smart contracts into three categories: Coding Conventions, General Vulnerability, Business Security. Their specific definitions are as follows:

- **Coding Conventions**

Audit whether smart contracts follow recommended language security coding practices. For example, smart contracts developed in Solidity language should fix the compiler version and do not use deprecated keywords.

- **General Vulnerability**

General Vulnerability include some common vulnerabilities that may appear in smart contract projects. These vulnerabilities are mainly related to the characteristics of the smart contract itself, such as integer overflow/underflow and denial of service attacks.

- **Business Security**

Business security is mainly related to some issues related to the business realized by each project, and has a relatively strong pertinence. For example, whether the lock-up plan in the code match the white paper, or the flash loan attack caused by the incorrect setting of the price acquisition oracle.

*Note that the project may suffer stake losses due to the integrated third-party protocol. This is not something Beosin can control. Business security requires the participation of the project party. The project party and users need to stay vigilant at all times.

3.3 Disclaimer

The Audit Report issued by Beosin is related to the services agreed in the relevant service agreement. The Project Party or the Served Party (hereinafter referred to as the "Served Party") can only be used within the conditions and scope agreed in the service agreement. Other third parties shall not transmit, disclose, quote, rely on or tamper with the Audit Report issued for any purpose.

The Audit Report issued by Beosin is made solely for the code, and any description, expression or wording contained therein shall not be interpreted as affirmation or confirmation of the project, nor shall any warranty or guarantee be given as to the absolute flawlessness of the code analyzed, the code team, the business model or legal compliance.

The Audit Report issued by Beosin is only based on the code provided by the Served Party and the technology currently available to Beosin. However, due to the technical limitations of any organization, and in the event that the code provided by the Served Party is missing information, tampered with, deleted, hidden or subsequently altered, the audit report may still fail to fully enumerate all the risks.

The Audit Report issued by Beosin in no way provides investment advice on any project, nor should it be utilized as investment suggestions of any type. This report represents an extensive evaluation process designed to help our customers improve code quality while mitigating the high risks in blockchain.

3.4 About Beosin

Beosin is the first institution in the world specializing in the construction of blockchain security ecosystem. The core team members are all professors, postdocs, PhDs, and Internet elites from world-renowned academic institutions. Beosin has more than 20 years of research in formal verification technology, trusted computing, mobile security and kernel security, with overseas experience in studying and collaborating in project research at well-known universities. Through the security audit and defense deployment of more than 2,000 smart contracts, over 50 public blockchains and wallets, and nearly 100 exchanges worldwide, Beosin has accumulated rich experience in security attack and defense of the blockchain field, and has developed several security products specifically for blockchain.

BEOSIN
Blockchain Security

Official Website

https://www.beosin.com

Telegram

https://t.me/+dD8Bnqd133RmNWNl

Twitter

https://twitter.com/Beosin_com

Email

Contact@beosin.com