Smart Contract Security Audit Report
For
Forsageant

Date Issued: Sep 26, 2023

Version: v1.0

Confidentiality Level: Public

Contents

1 Abstract

This report was prepared for Forsageant smart contract to identify issues and vulnerabilities in its smart contract source code. A thorough examination of Forsageant smart contracts was conducted through timely communication with Forsageant, static analysis using multiple audit tools and manual auditing of their smart contract source code.

The audit process paid particular attention to the following considerations.

- A thorough review of the smart contract logic flow
- Assessment of the code base to ensure compliance with current best practice and industry standards
- Ensured the contract logic met the client's specifications and intent
- Internal vulnerability scanning tools tested for common risks and writing errors
- Testing smart contracts for common attack vectors
- Test smart contracts for known vulnerability risks
- Conduct a thorough line-by-line manual review of the entire code base

As a result of the security assessment, issues ranging from critical to informational were identified. We recommend that these issues are addressed to ensure a high level of security standards and industry practice. The recommendations we made could have better served the project from a security perspective.

- Enhance general coding practices to improve the structure of the source code.
- Provide more comments for each function to improve readability.
- Provide more transparency of privileged activities once the agreement is in place.

2 Overview

2.1 Project Summary

Project Summary	Project Information
Name	Forsageant
Start date	Sep 20, 2023
End date	Sep 26, 2023
Contract type	DeFi
Language	Solidity
Commits	e62d10f12b4e3c24bf2e18a3af5667ba4d4d7e99
File	CardNFT.sol 、 FOLToken.sol 、 Family.sol、 Fomo.sol 、 Home.sol 、 Levels.sol 、 Migrations.sol、 Mine.sol 、 Nodes.sol 、 Random.sol 、 Week.sol 、 Achievement.sol 、 PermissionControl.sol 、 TranferEthWithCard.sol 、 UserInfo.sol 、 WrappedCoin.sol

2.2 Report HASH

Name	HASH
Forsageant	62E21F14510E21F9BE88C4E40711B5429B438300143A2A A10FDEB332E4FB50F2

3 Project contract details

3.1 Contract Overview

UserInfoLib.sol

Library contract, the main contract structure is UserInfo, the contract increasePendingReward method is mainly for the record of reward increase.

TranferEthWithCard.sol

There are two methods present in this contract, the __TranferEthOfCard_init method is used to modify the card address and the tranferEthTo method is used to return the funds receiving address from the card address and to send the funds.

Achievement.sol

The contract is mainly used to record user information and rewards information, childrenAchievementsOf is used to query the achievements of subordinates, distrubutionsForefathers to get the rank and tier rewards of superiors and write the rewards into the memory corresponding to the rewards, there are also two internal call methods in the contract, _increase is used to increase the quantity and _upgradeToLevel is used to upgrade the level.

CardNFT.sol

This contract mainly inherits the ERC721 standard contract, which is the NFT issuance portal, and its main functions include setting BaseUrl, casting NFTs, destroying NFTs, and transferring NFTs.

Family.sol

This contract mainly records the upper and lower levels of the tree, the functions include querying the choking superiors, getting the list of direct pushes, and adding superiors to the lower levels.

FOLToken.sol

The contract mainly issues ERC20 Token and its main function is to transfer money, which contains several variables that can be set by the privileged roles, such as Blocked, Guarded, Pair Updates, Buying and Selling Fees and Collecting Fee Addresses.

Home.sol

The project entry contract, which is initialized to determine Family, Levels, Mine, and AssetPool contracts, and whose main functions include depositing money, getting rewards, withdrawing funds, and getting access to a number of variables that can be set by upper and lower levels and privileged roles.

Mine.sol

The contract is mainly invoked by the Home contract, which contains the implementation of the project logic. The contract functions include calculating rewards, initializing liquidity, and obtaining rewards.

Nodes.sol

The contract consists of four main functions, the earned method where the user calculates the user's earnings, the takeReward method where the user obtains the user's rewards, the setNoderPowerDelegate method which is used to update multiple variables which is called by the Levels contract, and the distrubutionReward method which is used to update the user's rewards.

Random.sol

This contract is mainly used to calculate the random number, which is obtained through the seed method, which calculates the random number through block.timestamp, _seed, _internalRandomSeed, where the _seed variable can be updated through the update method of this contract.

3.2 Code Overview

Achievement Contract

Function Name	Visibility	Modifiers
__Achievement_init	Internal	onlyInitializing
levelOf	Public	-
childrenAchievementsOf	External	-
distrubutionsForefathers	External	-
_increase	Internal	-
_upgradeToLevel	Internal	-
whenLevelUpgraded	Internal	-

CardNFT Contract

Function Name	Visibility	Modifiers
initialize	External	initializer
ownerOfAddr	External	-
safeMint	External	-
setBaseUrl	External	onlyRole(DEFAULT_ADMIN_ROLE)
_beforeTokenTransfer	Internal	-
_burn	Internal	-
tokenURI	Public	-
supportsInterface	Public	-
_baseURI	Internal	-

Family Contract

Function Name	Visibility	Modifiers
initialize	Public	initializer
getForefathers	External	-
childrenOf	External	-
makeRelation	External	-
_makeRelationFrom	Internal	-

FOLToken Contract

Function Name	Visibility	Modifiers
addPair	External	onlyOwner
removePair	External	onlyOwner
setSellFee	External	onlyOwner
setBuyFee	External	onlyOwner
setBuyPreAddress	External	onlyOwner
setSellPreAddress	External	onlyOwner
addGuarded	External	onlyOwner
removeGuarded	External	onlyOwner
addBlocked	External	onlyOwner
removeBlocked	External	onlyOwner
clim	External	-
_transfer	Internal	-

PoolFomo Contract

Function Name	Visibility	Modifiers
initialize	Public	initializer
investHistoryLen	External	-
depositedDelegate	External	onlyRole(DELEGATE_ROLE)
restartTick	External	onlyRole(MANAGER_ROLE)
distrubtionPool	External	-

Home Contract

Function Name	Visibility	Modifiers
initialize	Public	initializer
setUnownedAssetTokenId	External	onlyRole(MANAGER_ROLE)
setWithdrawFeeReceiptor	External	onlyRole(MANAGER_ROLE)
setWithdrawFeeRatioAtLevel	External	onlyRole(MANAGER_ROLE)
_rewardIncreasedHandle	Internal	-
_getParentOfDeep	Internal	-
getAmountParams	Public	-
deposit	External	-
_deposit	Internal	-
earned	Public	-
takeReward	External	-
takeUnownedAssetBalance	External	-

Levels Contract

Function Name	Visibility	Modifiers
initialize	Public	initializer
setLevelRewardRatios	External	onlyRole(MANAGER_ROLE)
setLayerRewardRatios	External	onlyRole(MANAGER_ROLE)
setLayerRewardDepths	External	onlyRole(MANAGER_ROLE)
upgradeToLevel3	External	-
levelUpgrade	External	-
updateStartDelegate	External	onlyRole(DELEGATE_ROLE)
increaseDelegate	External	onlyRole(DELEGATE_ROLE)
whenLevelUpgraded	Internal	-

Mine Contract

Function Name	Visibility	Modifiers
initialize	Public	initializer
addPowerDelegate	External	onlyRole(DELEGATE_ROLE)
clearPowerDelegate	External	onlyRole(DELEGATE_ROLE)
setMineEnable	External	onlyRole(MANAGER_ROLE)
initLiquidity	External	onlyRole(MANAGER_ROLE)
_getMaxOutput	Internal	-
_pendingAccountPerShare	Internal	-
updateAccountPerShare	External	-
earned	Public	-
_rewardTokenPerCurrency	Internal	-
takeReward	External	-

Nodes Contract

Function Name	Visibility	Modifiers
initialize	Public	initializer
earned	Public	-
takeReward	External	-
setNoderPowerDelegate	External	onlyRole(DELEGATE_ROLE)
distrubutionReward	External	-

Nodes Contract

Function Name	Visibility	Modifiers
initialize	Public	initializer
currentIndex	External	-
topAddressInfoAtEpoch	External	-
distrubtionAtEpoch	External	-
depositedDelegate	External	onlyRole(DELEGATE_ROLE)

4 Audit results

4.1 Key messages

ID	Title	Severity	Status
01	Controllable seed() method return value	Low	Confirmed
02	NFT baseUrl that can be modified	Informational	Confirmed
03	The distrubtionPool method may not be called normally	Low	Confirmed
04	Flash loan controllable rewardTokenPrice price	Low	Confirmed
05	Any address that exists can receive rewarded cardAddr address rewards	High	Fixed
06	Front-running transactions can receive more rewards	Low	Confirmed
07	Possible handling fees that may be too large	Informational	Confirmed
08	Users on the blacklist may be unable to trade	Informational	Confirmed
09	Transfer fees may be sent to the trading pair address	Informational	Confirmed

4.2 Audit details

4.2.1 Controllable seed() method return value

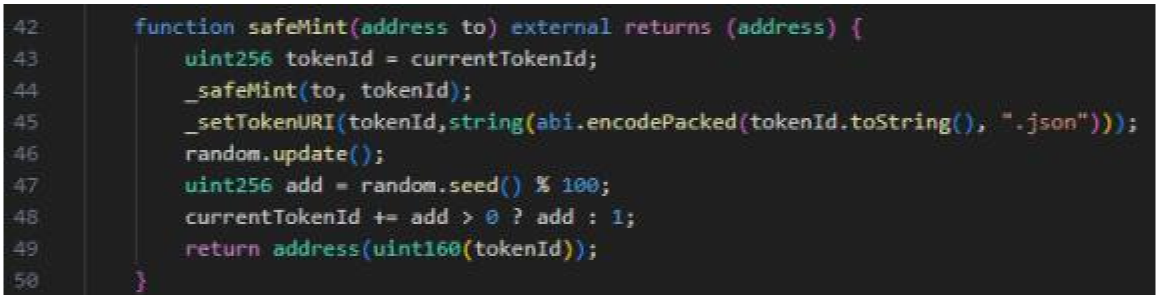
ID	Severity	Location	Status
01	Low	CardNFT.sol: 43, 50 random.sol: 16, 22	confirmed

Description

The safeMint() method uses random.seed() to fetch the value and compute it in that contract, due to the fact that random parameters are used in the computation in the seed() method and the _seed variable can be adjusted arbitrarily. If the random.seed() value is fetched in advance, it will result in the add variable being controllable.

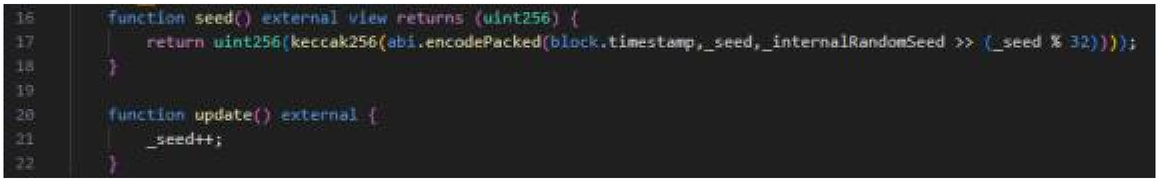
Code location:

CardNFT.sol



```
42      function safeMint(address to) external returns (address) {
43          uint256 tokenId = currentTokenId;
44          _safeMint(to, tokenId);
45          _setTokenURI(tokenId,string(abi.encodePacked(tokenId.toString(), ".json")));
46          random.update();
47          uint256 add = random.seed() % 100;
48          currentTokenId += add > 0 ? add : 1;
49          return address(uint160(tokenId));
50      }
```

random.sol



```
16      function seed() external view returns (uint256) {
17          return uint256(keccak256(abi.encodePacked(block.timestamp,_seed,_internalRandomSeed >> (_seed % 32))));
18      }
19
20      function update() external {
21          _seed++;
22      }
```

Recommendation

Using uncontrollable parameters as random numbers.

Status

Confirmed.

random.seed() is to get a random number The return value of this method is itself a random number and is not controllable (theoretically controllable, but infinitely expensive to manipulate).

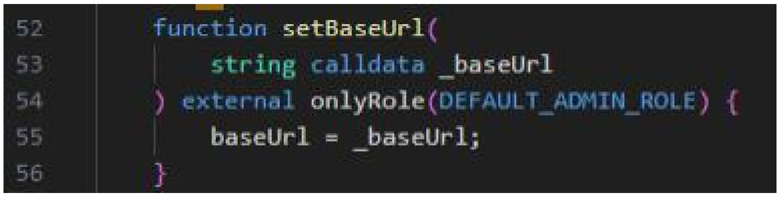
4.2.2 NFT baseUrl that can be modified

ID	Severity	Location	Status
02	Informational	CardNFT.sol: 52,56;	confirmed

Description

The DEFAULT_ADMIN_ROLE privileged role can set baseUrl. If DEFAULT_ADMIN_ROLE is a privileged role, it is EOA. If EOA is maliciously controlled, it may affect the file storage location of the entire NFT.

Code location:



```
52      function setBaseUrl(
53          string calldata _baseUrl
54      ) external onlyRole(DEFAULT_ADMIN_ROLE) {
55          baseUrl = _baseUrl;
56      }
```

Recommendation

It is recommended that privileged roles be managed using multi-signature or time locks.

Status

Confirmed.

baseUrl business requirements are like this.

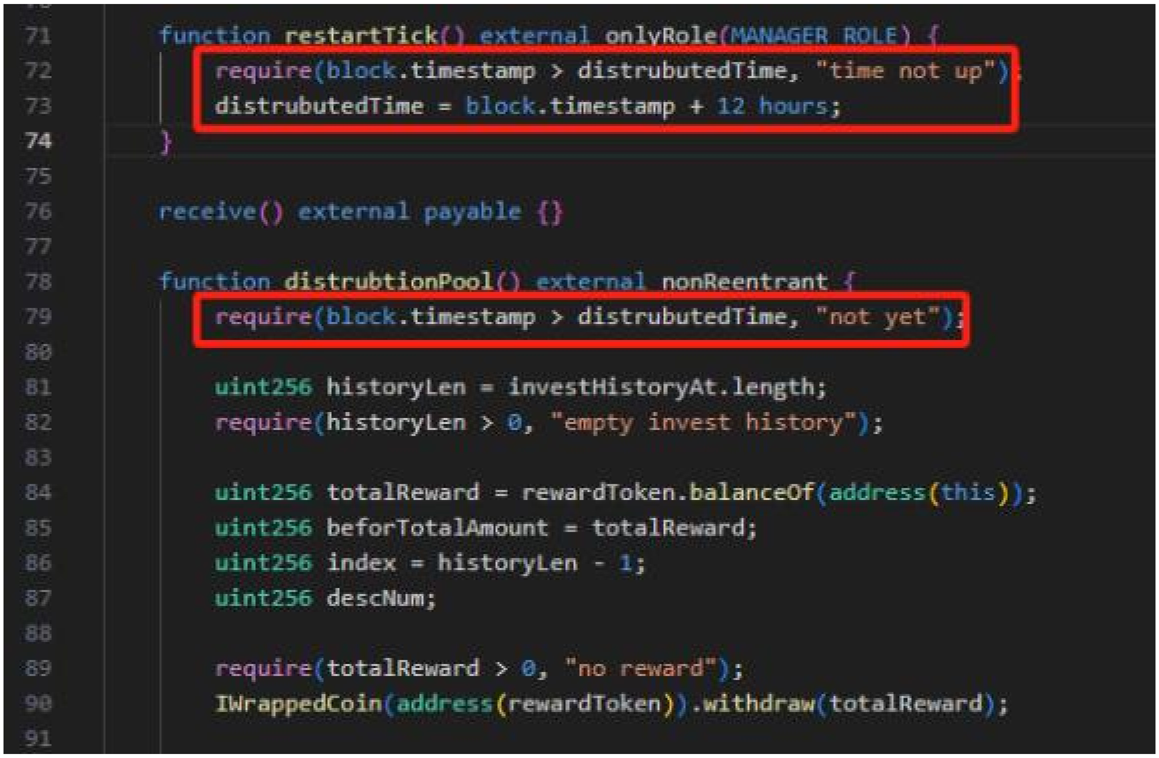
4.2.3 The distrubtionPool method may not be called normally

ID	Severity	Location	Status
03	Low	Fomo.sol: 71, 91	confirmed

Description

The restartTick method is used to update the distrubutedTime variable. Since distrubutedTime can be updated all the time, there may be distrubutedTime updates that cause the block.timestamp > distrubutedTime condition in the distrubtionPool method to fail, and the distrubtionPool method cannot be called normally.

Code location:



```
71      function restartTick() external onlyRole(MANAGER_ROLE) {
72          require(block.timestamp > distrubutedTime, "time not up");
73          distrubutedTime = block.timestamp + 12 hours;
74      }
75
76      receive() external payable {}
77
78      function distrubtionPool() external nonReentrant {
79          require(block.timestamp > distrubutedTime, "not yet");
80
81          uint256 historyLen = investHistoryAt.length;
82          require(historyLen > 0, "empty invest history");
83
84          uint256 totalReward = rewardToken.balanceOf(address(this));
85          uint256 beforTotalAmount = totalReward;
86          uint256 index = historyLen - 1;
87          uint256 descNum;
88
89          require(totalReward > 0, "no reward");
90          IWrappedCoin(address(rewardToken)).withdraw(totalReward);
91
```

Recommendation

It is recommended to change the modification range of the distributedTime variable to ensure the normal execution of the distrubtionPool method.

Status

Confirmed.

The business of distrubutedTime is that restartTick() needs to be called only when distrubutedTime does not change for a long time. This method is for disaster recovery.

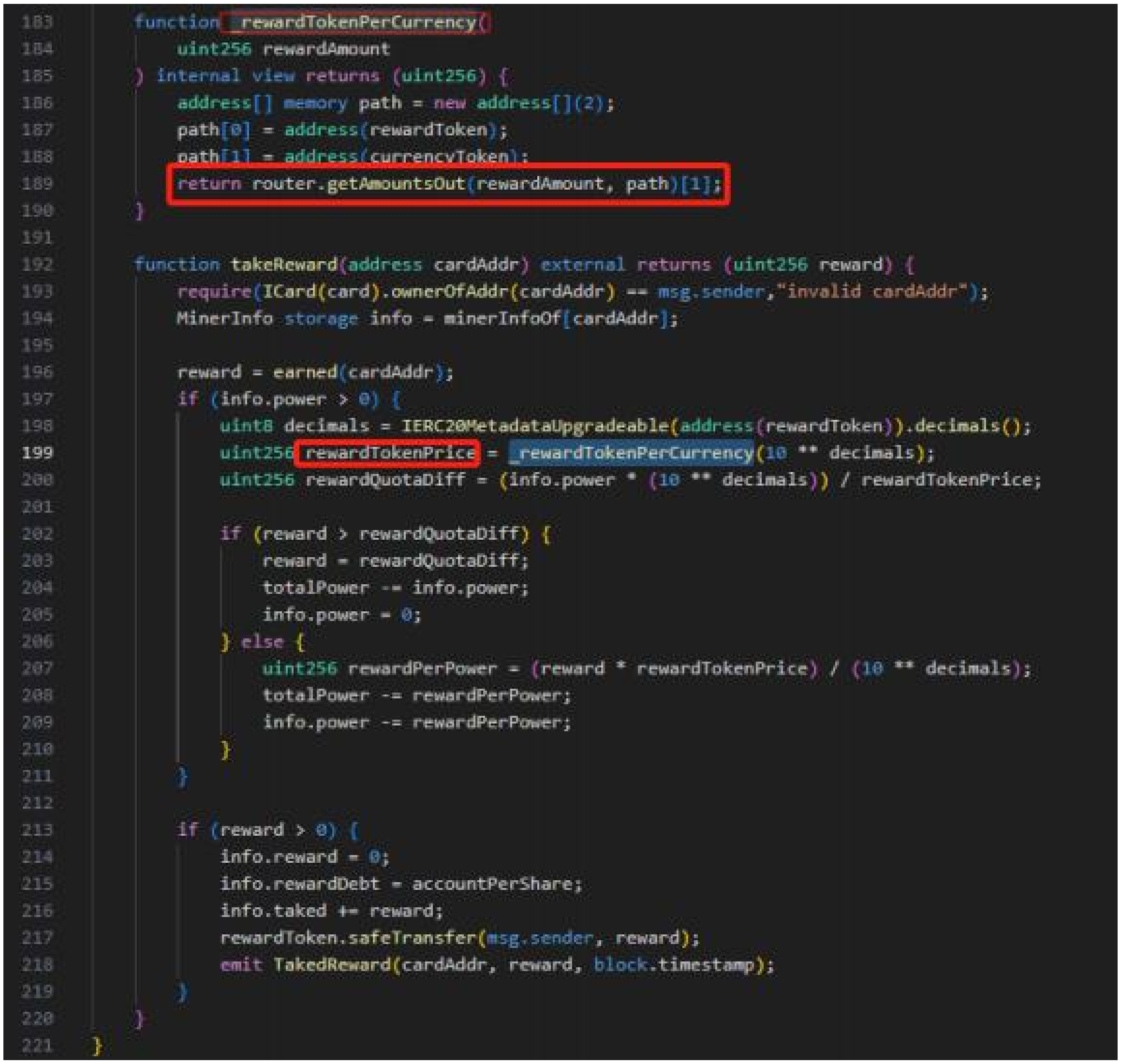
4.2.4 Flash loan controllable rewardTokenPrice price

ID	Severity	Location	Status
04	Low	Mine.sol: 183, 221	confirmed

Description

The takeReward method allows users to obtain rewards, and the price of the reward token, rewardTokenPrice, is obtained through the real-time price of the transaction. The price of the trading pair can be controlled through flash loans, thereby controlling the rewardTokenPrice price, and ultimately affecting the user reward through rewardTokenPrice calculation.

Code location:



```
183     function _rewardTokenPerCurrency(
184         uint256 rewardAmount
185     ) internal view returns (uint256) {
186         address[] memory path = new address[](2);
187         path[0] = address(rewardToken);
188         path[1] = address(currencyToken);
189         return router.getAmountsOut(rewardAmount, path)[1];
190     }
191
192     function takeReward(address cardAddr) external returns (uint256 reward) {
193         require(ICard(card).ownerOfAddr(cardAddr) == msg.sender,"invalid cardAddr");
194         MinerInfo storage info = minerInfoOf[cardAddr];
195
196         reward = earned(cardAddr);
197         if (info.power > 0) {
198             uint8 decimals = IERC20MetadataUpgradeable(address(rewardToken)).decimals();
199             uint256 rewardTokenPrice = _rewardTokenPerCurrency(10 ** decimals);
200             uint256 rewardQuotaDiff = (info.power * (10 ** decimals)) / rewardTokenPrice;
201
202             if (reward > rewardQuotaDiff) {
203                 reward = rewardQuotaDiff;
204                 totalPower -= info.power;
205                 info.power = 0;
206             } else {
207                 uint256 rewardPerPower = (reward * rewardTokenPrice) / (10 ** decimals);
208                 totalPower -= rewardPerPower;
209                 info.power -= rewardPerPower;
210             }
211         }
212
213         if (reward > 0) {
214             info.reward = 0;
215             info.rewardDebt = accountPerShare;
216             info.taked += reward;
217             rewardToken.safeTransfer(msg.sender, reward);
218             emit TakedReward(cardAddr, reward, block.timestamp);
219         }
220     }
221 }
```

Recommendation

It is recommended to use the weighted average price or external reliable price interface when obtaining prices.

Status

Confirmed.

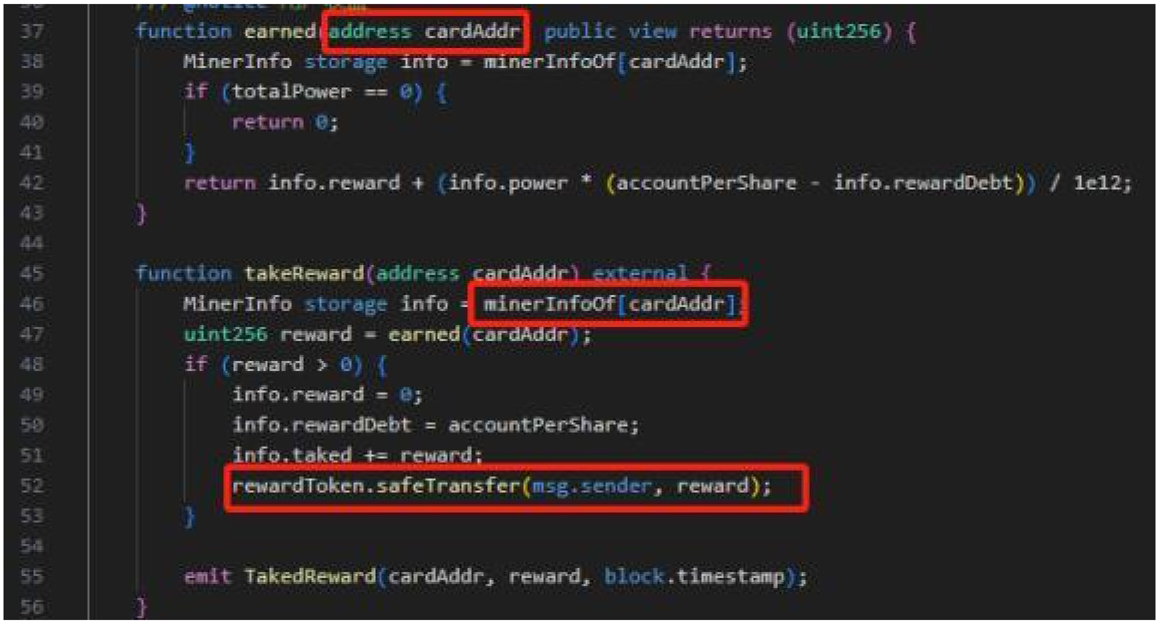
Flash loans only exist in theory. In actual operations, attacks cannot directly obtain benefits. Therefore, grade operations cannot cover the cost of flash loans.

4.2.5 Any address that exists can receive rewarded cardAddr address rewards

ID	Severity	Location	Status
05	High	Nodes.sol: 37, 56	fixed

Description

The takeReward method allows users to obtain cardAddr address rewards, but this method does not determine the relationship between cardAddr and msg.sender addresses, so any address can enter a rewarded cardAddr address to receive rewards.



```
37      function earned(address cardAddr) public view returns (uint256) {
38          MinerInfo storage info = minerInfoOf[cardAddr];
39          if (totalPower == 0) {
40              return 0;
41          }
42          return info.reward + (info.power * (accountPerShare - info.rewardDebt)) / 1e12;
43      }
44
45      function takeReward(address cardAddr) external {
46          MinerInfo storage info = minerInfoOf[cardAddr];
47          uint256 reward = earned(cardAddr);
48          if (reward > 0) {
49              info.reward = 0;
50              info.rewardDebt = accountPerShare;
51              info.taked += reward;
52              rewardToken.safeTransfer(msg.sender, reward);
53          }
54
55          emit TakedReward(cardAddr, reward, block.timestamp);
56      }
```

Recommendation

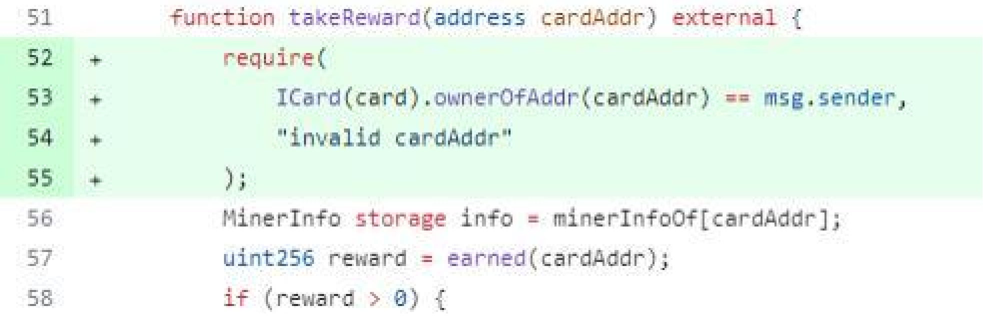
It is recommended to determine the relationship between cardAddr and msg.sender addresses to avoid receiving rewards at any address.

Status

Fixed.

The takeReward() method can indeed receive rewards from others. This is a major BUG. Fixed and added NFT holder determination.

https://github.com/forsageant/contracts/commit/3f6aa967242c4de5aeb02ac2f4bbf32f1b43f917



```
      function takeReward(address cardAddr) external {
+         require(
+             ICard(card).ownerOfAddr(cardAddr) == msg.sender,
+             "invalid cardAddr"
+         );
          MinerInfo storage info = minerInfoOf[cardAddr];
          uint256 reward = earned(cardAddr);
          if (reward > 0) {
```

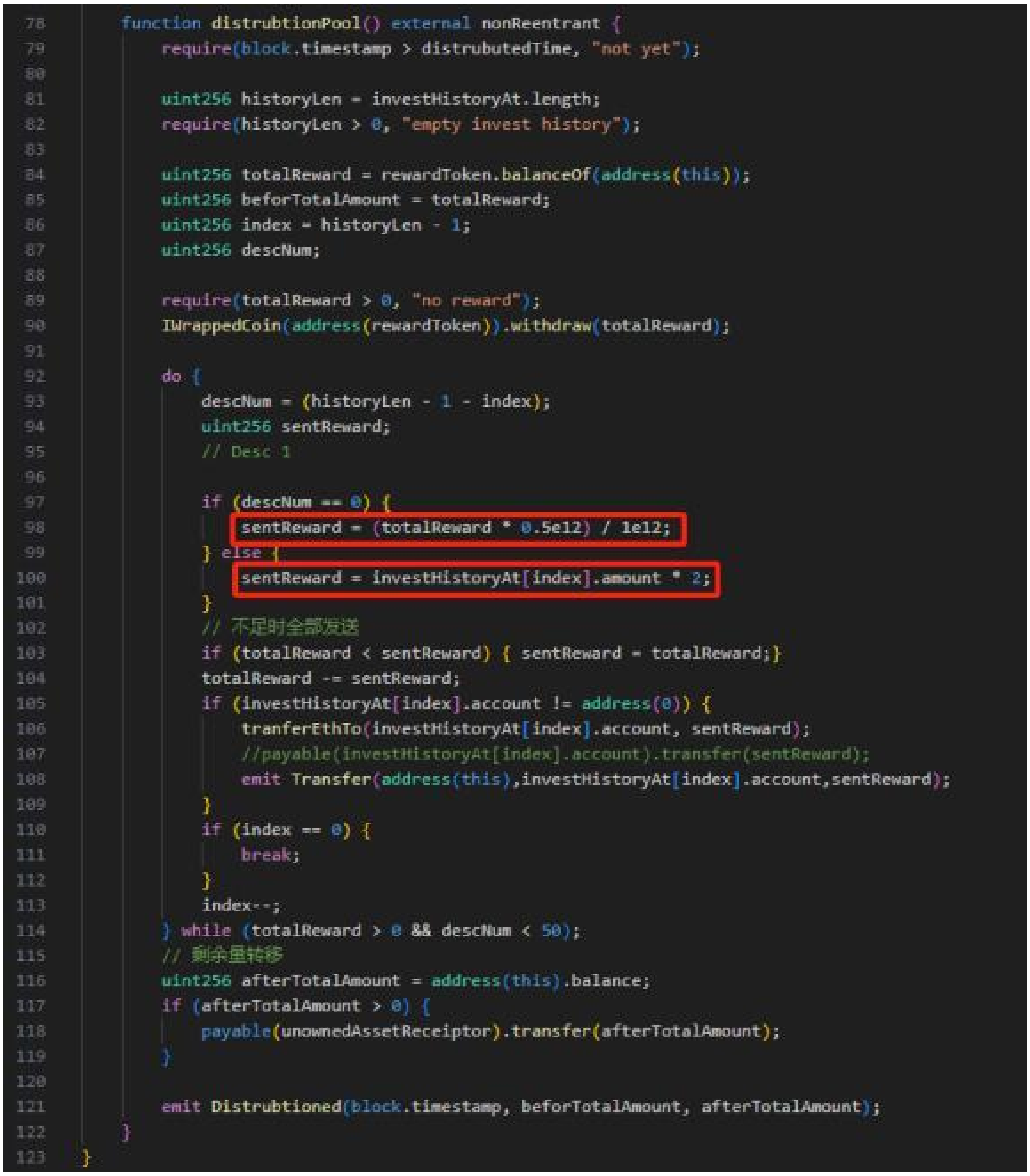
4.2.6 Front-running transactions can receive more rewards

ID	Severity	Location	Status
06	Low	Fomo.sol: 78, 123	confirmed

Description

The distrubtionPool method has two ways to calculate the sent reward sentReward. The first is half of the current total reward, and the second is calculated based on the current amount. If the first type of reward is large, the attacker can use front-running transactions to obtain the first type of reward calculation every time it reaches the first position of investHistoryAt.

Code location:



```
    function distrubtionPool() external nonReentrant {
        require(block.timestamp > distrubutedTime, "not yet");

        uint256 historyLen = investHistoryAt.length;
        require(historyLen > 0, "empty invest history");

        uint256 totalReward = rewardToken.balanceOf(address(this));
        uint256 beforTotalAmount = totalReward;
        uint256 index = historyLen - 1;
        uint256 descNum;

        require(totalReward > 0, "no reward");
        IWrappedCoin(address(rewardToken)).withdraw(totalReward);

        do {
            descNum = (historyLen - 1 - index);
            uint256 sentReward;
            // Desc 1

            if (descNum == 0) {
                sentReward = (totalReward * 0.5e12) / 1e12;
            } else {
                sentReward = investHistoryAt[index].amount * 2;
            }
            // 不足时全部发送
            if (totalReward < sentReward) { sentReward = totalReward;}
            totalReward -= sentReward;
            if (investHistoryAt[index].account != address(0)) {
                tranferEthTo(investHistoryAt[index].account, sentReward);
                //payable(investHistoryAt[index].account).transfer(sentReward);
                emit Transfer(address(this),investHistoryAt[index].account,sentReward);
            }
            if (index == 0) {
                break;
            }
            index--;
        } while (totalReward > 0 && descNum < 50);
        // 剩余量转移
        uint256 afterTotalAmount = address(this).balance;
        if (afterTotalAmount > 0) {
            payable(unownedAssetReceiptor).transfer(afterTotalAmount);
        }

        emit Distrubtioned(block.timestamp, beforTotalAmount, afterTotalAmount);
    }
}
```

Recommendation

Calculate all rewards by the current amount.

Status

Confirmed.

Business needs.

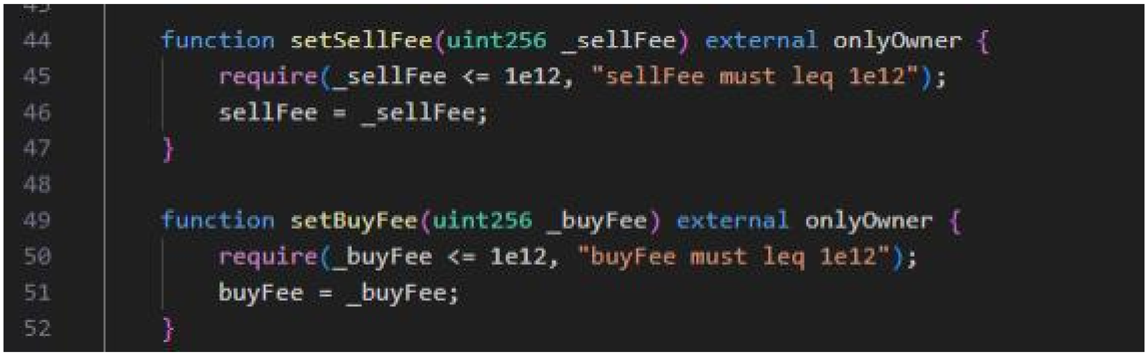
4.2.7 Possible handling fees that may be too large

ID	Severity	Location	Status
07	Informational	FOLToken.sol: 44, 52	confirmed

Description

The setSellFee() method and the setBuyFee() method are called by the owner privileged role. These two methods are used to modify the handling fees for buying and selling tokens respectively. If the handling fee rate is large, it may cause the user's principal to be damaged, and ultimately In the extreme case, when the transaction fee is set to 1, all funds transferred by the user will be deducted as fees.

Code location:



```
44      function setSellFee(uint256 _sellFee) external onlyOwner {
45          require(_sellFee <= 1e12, "sellFee must leq 1e12");
46          sellFee = _sellFee;
47      }
48
49      function setBuyFee(uint256 _buyFee) external onlyOwner {
50          require(_buyFee <= 1e12, "buyFee must leq 1e12");
51          buyFee = _buyFee;
52      }
```

Recommendation

It is recommended that the transaction fee set by judgment should not exceed 20%.

Status

Confirmed.

Business needs.

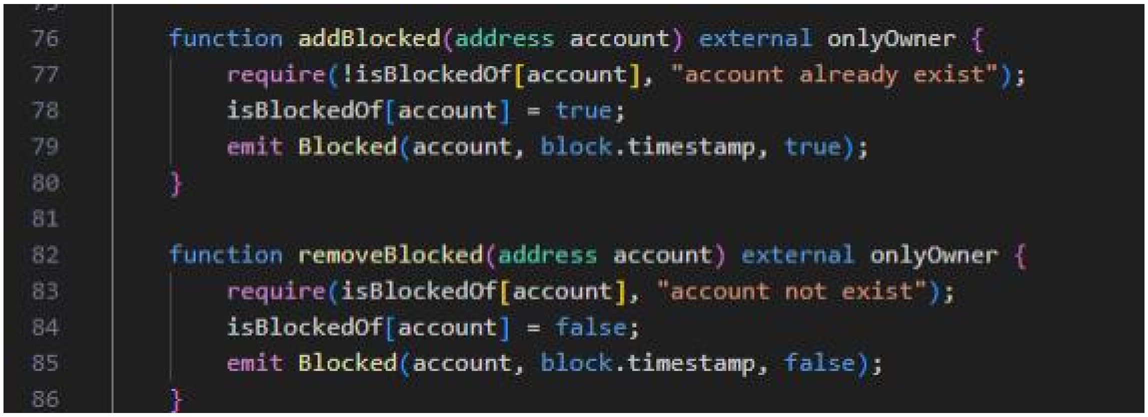
4.2.8 Users on the blacklist may be unable to trade

ID	Severity	Location	Status
08	Informational	FOLToken.sol: 76, 86	confirmed

Description

The addBlocked() method and removeBlocked() method are called by the owner privileged role. These two methods are used to modify whether the user is a blacklisted address. Once the user's address is set to the blacklist, the user's funds will not be able to trade normally.

Code location:



```
76      function addBlocked(address account) external onlyOwner {
77          require(!isBlockedOf[account], "account already exist");
78          isBlockedOf[account] = true;
79          emit Blocked(account, block.timestamp, true);
80      }
81
82      function removeBlocked(address account) external onlyOwner {
83          require(isBlockedOf[account], "account not exist");
84          isBlockedOf[account] = false;
85          emit Blocked(account, block.timestamp, false);
86      }
```

Recommendation

It is recommended to remove the blacklist restriction.

Status

Confirmed.

Business needs.

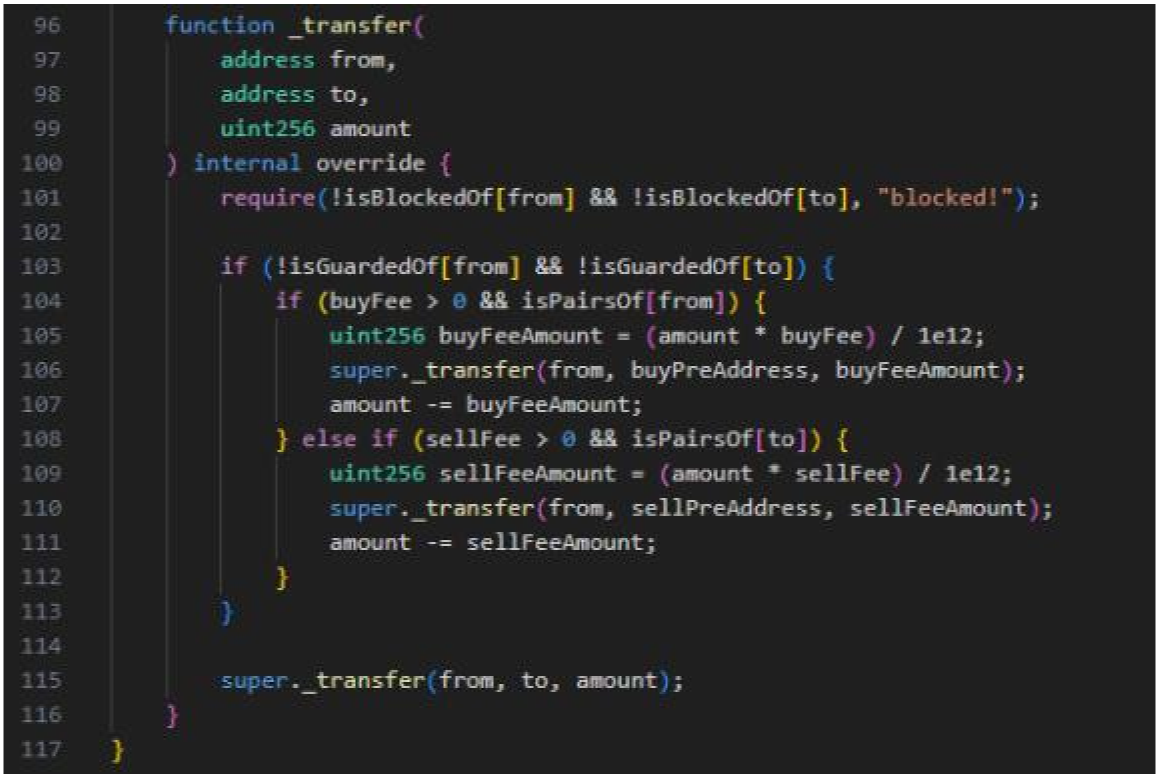
4.2.9Transfer fees may be sent to the trading pair address

ID	Severity	Location	Status
09	Informational	FOLToken.sol: 96, 117	confirmed

Description

The _transfer() method is used for user transfers. In this method, the transaction pair address will be judged, and the transaction fee will be sent to the corresponding address. The transfer fee collection address cannot be the transaction pair address. If it is the transaction pair address, it will cause The trading pair experiences price fluctuations.

Code location:



```
96      function _transfer(
97          address from,
98          address to,
99          uint256 amount
100     ) internal override {
101         require(!isBlockedOf[from] && !isBlockedOf[to], "blocked!");
102 
103         if (!isGuardedOf[from] && !isGuardedOf[to]) {
104             if (buyFee > 0 && isPairsOf[from]) {
105                 uint256 buyFeeAmount = (amount * buyFee) / 1e12;
106                 super._transfer(from, buyPreAddress, buyFeeAmount);
107                 amount -= buyFeeAmount;
108             } else if (sellFee > 0 && isPairsOf[to]) {
109                 uint256 sellFeeAmount = (amount * sellFee) / 1e12;
110                 super._transfer(from, sellPreAddress, sellFeeAmount);
111                 amount -= sellFeeAmount;
112             }
113         }
114 
115         super._transfer(from, to, amount);
116     }
117 }
```

Recommendation

It is recommended to limit the trading fee address to a trading pair address.

Status

Confirmed.

Business needs.

5 Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in-storage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Disclaimer

This report is issued in response to facts that occurred or existed prior to the issuance of this report, and liability is assumed only on that basis.
Shield Security cannot determine the security status of this program and assumes no responsibility for facts occurring or existing after the date of this report. The security audit analysis and other content in this report is based on documents and materials provided to Shield Security by the information provider through the date of the insurance report. in Shield Security's opinion. The information provided is not missing, falsified, deleted or concealed. If the information provided is missing, altered, deleted, concealed or not in accordance with the actual circumstances, Shield Security shall not be liable for any loss or adverse effect resulting therefrom. shield Security will only carry out the agreed security audit of the security status of the project and issue this report. shield Security is not responsible for the background and other circumstances of the project. Shield Security is not responsible for the background and other circumstances of the project.