Meta Smart Contract

February 2024

SMART CONTRACT AUDIT REPORT

ExVul

www.exvul.com

Table of Contents

1. EXECUTIVE SUMMARY

Exvul Web3 Security was engaged by **Meta** to review smart contract implementation. The assessment was conducted in accordance with our systematic approach to evaluate potential security issues based upon customer requirement. The report provides detailed recommendations to resolve the issue and provide additional suggestions or recommendations for improvement.

High risk finding is primarily related to the fee issue and functions call permission issue.

Low risk findings are primarily related to NFT link.

Informational risk finding is primarily related to the redundant code.

The outcome of the assessment outlined in chapter 3 provides the system's owners a full description of the vulnerabilities identified, the associated risk rating for each vulnerability, and detailed recommendations that will resolve the underlying technical issue.

1.1 Methodology

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology [10] which is the gold standard in risk assessment using the following risk models:

- Likelihood: represents how likely a particular vulnerability is to be uncovered and exploited in the wild.
- Impact: measures the technical loss and business damage of a successful attack.
- Severity: determine the overall criticality of the risk.

Likelihood can be: High, Medium and Low and impact are categorized into for: High, Medium, Low, Informational. Severity is determined by likelihood and impact and can be classified into five categories accordingly, Critical, High, Medium, Low, Informational shown in table 1.1.

Likelihood	Informational	Low	Medium	High
High	*Informational*	Medium	High	Critical
Medium	*Informational*	Low	Medium	High
Low	*Informational*	Low	Low	Medium

IMPACT

Table 1.1 Overall Risk Severity

To evaluate the risk, we will be going through a list of items, and each would be labelled with a severity category. The audit was performed with a systematic approach guided by a comprehensive assessment list carefully designed to identify known and impactful security issues. If our tool or analysis does not identify any issue, the contract can be considered safe regarding the assessed item. For any discovered issue, we might further deploy contracts on our private test environment and run tests to confirm the findings. If necessary, we would additionally build a PoC to demonstrate the possibility of exploitation. The concrete list of check items is shown in Table 1.2.

- Basic Coding Bugs: We first statically analyze given smart contracts with our proprietary static code analyzer for known coding bugs, and then manually verify (reject or confirm) all the issues found by our tool.
- Code and business security testing: We further review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.
- Additional Recommendations: We also provide additional suggestions regarding the coding and development of smart contracts from the perspective of proven programming practices.

Category	Assessment Item
Basic Coding Assessment	Apply Verification Control
	Authorization Access Control
	Forged Transfer Vulnerability
	Forged Transfer Notification
	Numeric Overflow
	Transaction Rollback Attack
	Transaction Block Stuffing Attack
	Soft Fail Attack
	Hard Fail Attack
	Abnormal Memo
	Abnormal Resource Consumption
	Secure Random Number
Advanced Source Code Scrutiny	Asset Security
	Cryptography Security
	Business Logic Review
	Source Code Functional Verification
	Account Authorization Control
	Sensitive Information Disclosure
	Circuit Breaker

Category	Assessment Item
	Blacklist Control
	System API Call Analysis
	Contract Deployment Consistency Check
Additional Recommendations	Semantic Consistency Checks
	Following Other Best Practices

Table 1.2: The Full List of Assessment Items

To better describe each issue we identified, we categorize the findings with Common Weakness Enumeration (CWE-699) [14], which is a community-developed list of software weakness types to better delineate and organize weaknesses around concepts frequently encountered in software development.

2. FINDINGS OVERVIEW

2.1 Project Info And Contract Address

Project Name: Meta

Audit Time: February26nd, 2024 – Mar5th, 2024

Language: solidity

File Name	MD5
meta-main	36FE4F80C05B122B3FFA603AC6822C94

2.2 Summary

Severity	Found	
Critical	0	
High	2	🟥🟥
Medium	5	🟨🟨🟨🟨🟨
Low	1	🟦
Informational	1	🟩

2.3 Key Findings

High risk finding is primarily related to the fee issue and functions call permission issue.

Low risk findings are primarily related to NFT link.

Informational risk finding is primarily related to the redundant code.

ID	Severity	Findings Title	Status	Confirm
NVE- 001	High	Fee issue	Ignored	Confirmed
NVE- 002	High	Functions call permission issue	Ignored	Confirmed
NVE- 003	Medium	Privileged role	Ignored	Confirmed
NVE- 004	Medium	Lock token issue	Ignored	Confirmed
NVE- 005	Medium	Time setting logic issue	Ignored	Confirmed
NVE- 006	Medium	Random numbers are predicted	Ignored	Confirmed
NVE- 007	Medium	withdrawInviteReward function	Ignored	Confirmed
NVE- 008	Low	NFT URI issue	Ignored	Confirmed
NVE- 009	Informational	Redundant code	Ignored	Confirmed

Table 2.1: Key Audit Findings

3. DETAILED DESCRIPTION OF FINDINGS

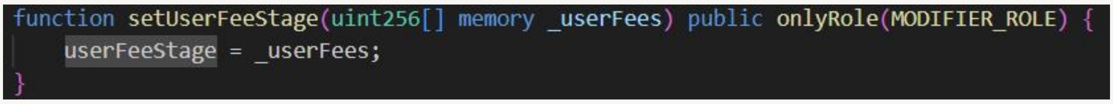
3.1 Fee issue

ID:	NVE-001	**Location:**	Farm.sol,RoBoExchange.sol , SwapFee.sol
Severity:	High	**Category:**	Business Issues
Likelihood:	High	**Impact:**	High

Description:

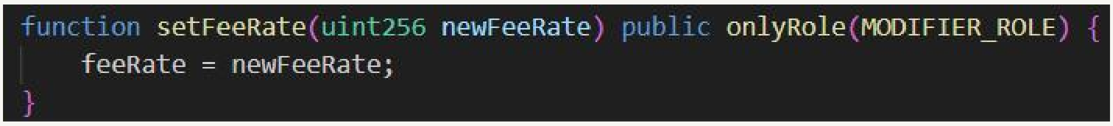
As shown in the figure below, the relevant functions for setting fees in the Farm.sol, RoBoExchange.sol, and SwapFee.sol contracts do not set a fee limit.

- If the fee of the Farm.sol contract is too high, it will affect the user's staked assets. When the user withdraws the staked assets, all the assets can be used as handling fees.
- If the handling fee in the RoBoExchange.sol contract is set too high, it will affect the assets of the order holder and may result in all assets being handling fees.
- The fee set in the SwapFee.sol contract will affect the fee calculation.



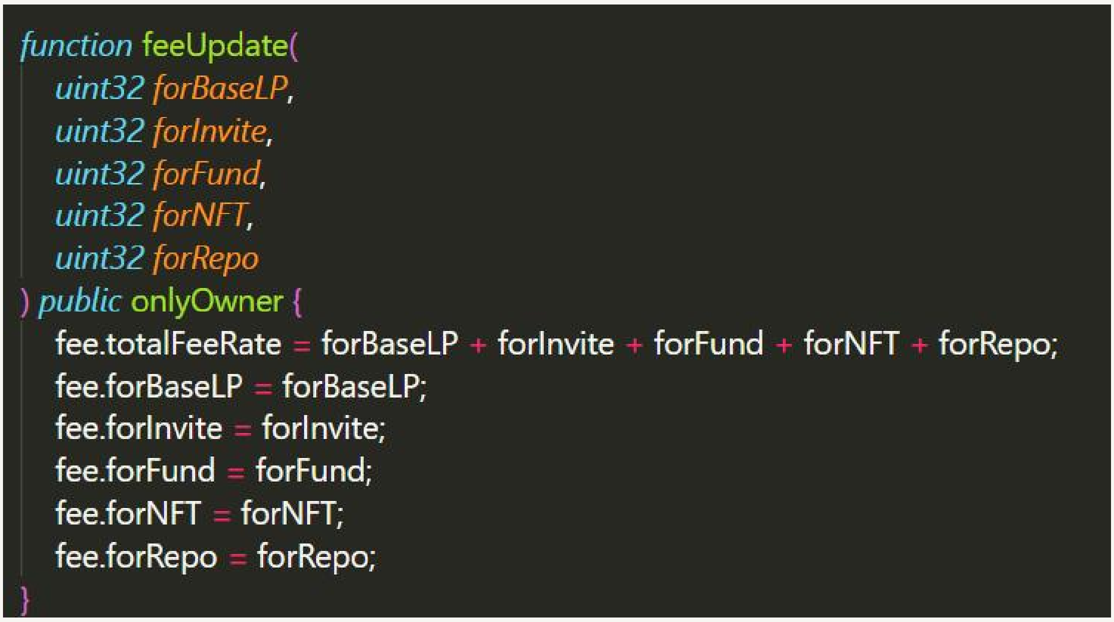
```
function setUserFeeStage(uint256[] memory _userFees) public onlyRole(MODIFIER_ROLE) {
    userFeeStage = _userFees;
}
```

Figure 3.1.1 Farm.sol



```
function setFeeRate(uint256 newFeeRate) public onlyRole(MODIFIER_ROLE) {
    feeRate = newFeeRate;
}
```

Figure 3.1.2 RoBoExchange.sol



```
function feeUpdate(
    uint32 forBaseLP,
    uint32 forInvite,
    uint32 forFund,
    uint32 forNFT,
    uint32 forRepo
) public onlyOwner {
    fee.totalFeeRate = forBaseLP + forInvite + forFund + forNFT + forRepo;
    fee.forBaseLP = forBaseLP;
    fee.forInvite = forInvite;
    fee.forFund = forFund;
    fee.forNFT = forNFT;
    fee.forRepo = forRepo;
}
```

Figure 3.1.3 SwapFee.sol

Recommendations:

ExVul Web3 Labs recommends setting a fee limit.

Result: Confirmed

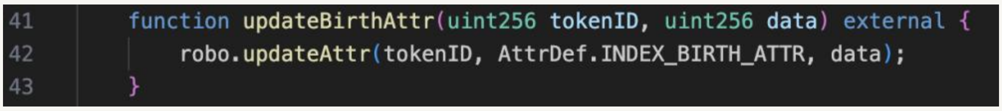
Fix Result: Ignored

3.2 Functions call permission issue

ID:	NVE-002	**Location:**	AttributeManager.sol
Severity:	High	**Category:**	Business Issues
Likelihood:	High	**Impact:**	High

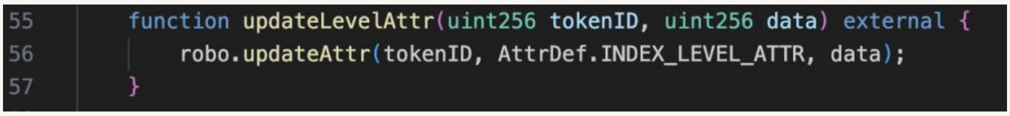
Description:

If the AttributeManager.Sol contract is set to true of the hasContract mapping in the core.sol contract, any user can call the updateBirthAttr function and the updateLevelAttr function to update the attributes in the robo contract.



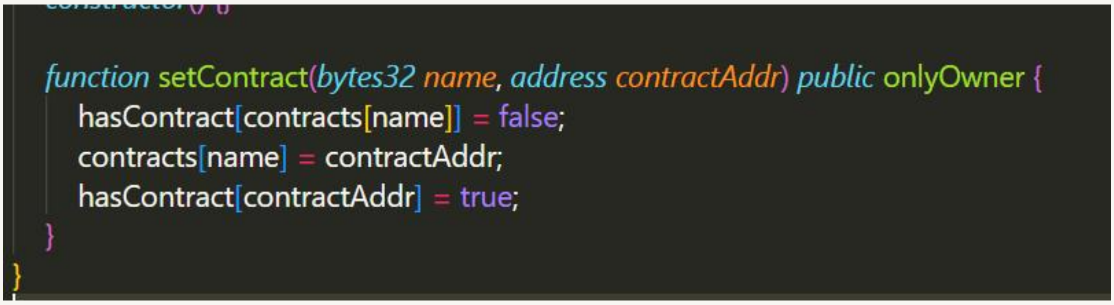
```
    function updateBirthAttr(uint256 tokenID, uint256 data) external {
        robo.updateAttr(tokenID, AttrDef.INDEX_BIRTH_ATTR, data);
    }
```

Figure 3.2.1 AttributeManager.sol



```
    function updateLevelAttr(uint256 tokenID, uint256 data) external {
        robo.updateAttr(tokenID, AttrDef.INDEX_LEVEL_ATTR, data);
    }
```

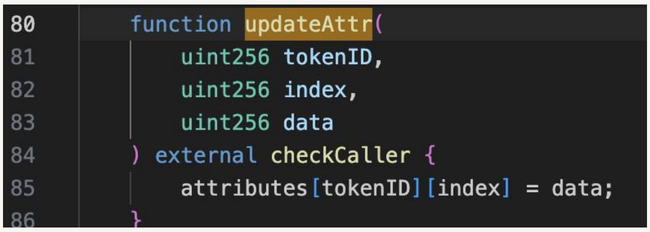
Figure 3.2.2 AttributeManager.sol



```
    constructor() {}

    function setContract(bytes32 name, address contractAddr) public onlyOwner {
        hasContract[contracts[name]] = false;
        contracts[name] = contractAddr;
        hasContract[contractAddr] = true;
    }
}
```

Figure 3.2.3 core.sol



```
function updateAttr(
    uint256 tokenID,
    uint256 index,
    uint256 data
) external checkCaller {
    attributes[tokenID][index] = data;
}
```

Figure 3.2.3 robo.sol

Recommendations:

ExVul Web3 Labs recommends adding permissions for functions to be called..

Result: Confirmed

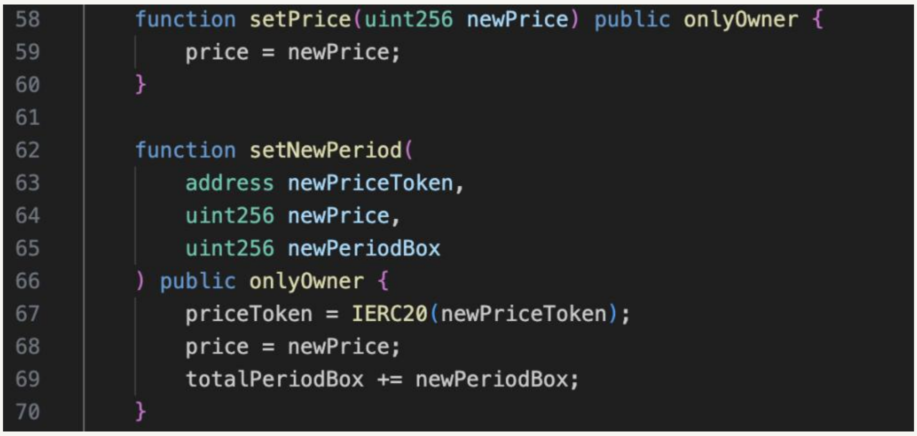
Fix Result: Ignore

3.3 Privileged role

ID:	NVE-003	**Location:**	AirBox.sol，AirWhiteBox.sol
Severity:	Low	**Category:**	Business Issues
Likelihood:	Low	**Impact:**	Medium

Description:

- The owenr of the AirBox.sol contract can call the setPrice function and setNewPeriod function to set the price and period, which affects the price and period of user purchases.

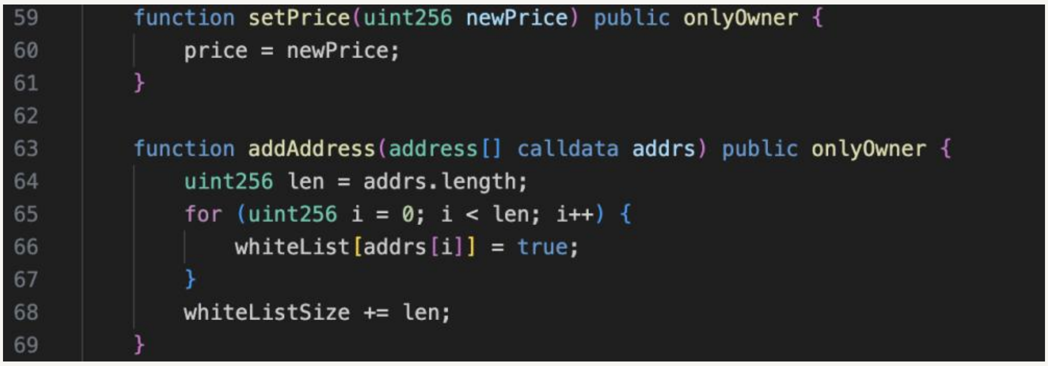


```
function setPrice(uint256 newPrice) public onlyOwner {
    price = newPrice;
}

function setNewPeriod(
    address newPriceToken,
    uint256 newPrice,
    uint256 newPeriodBox
) public onlyOwner {
    priceToken = IERC20(newPriceToken);
    price = newPrice;
    totalPeriodBox += newPeriodBox;
}
```

Figure 3.3.1 AirBox.sol

- The AirWhiteBox.sol contract owner can call the setPrice function and addAddress function to set the price and add whitelist addresses, which affects the user's buy price.



```
    function setPrice(uint256 newPrice) public onlyOwner {
        price = newPrice;
    }

    function addAddress(address[] calldata addrs) public onlyOwner {
        uint256 len = addrs.length;
        for (uint256 i = 0; i < len; i++) {
            whiteList[addrs[i]] = true;
        }
        whiteListSize += len;
    }
```

Figure 3.3.2 AirBox.sol

Recommendations:

ExVul Web3 Labs recommends the contract owner is managed using multi-signatures.

Result: Confirmed

Fix Result: Ignored

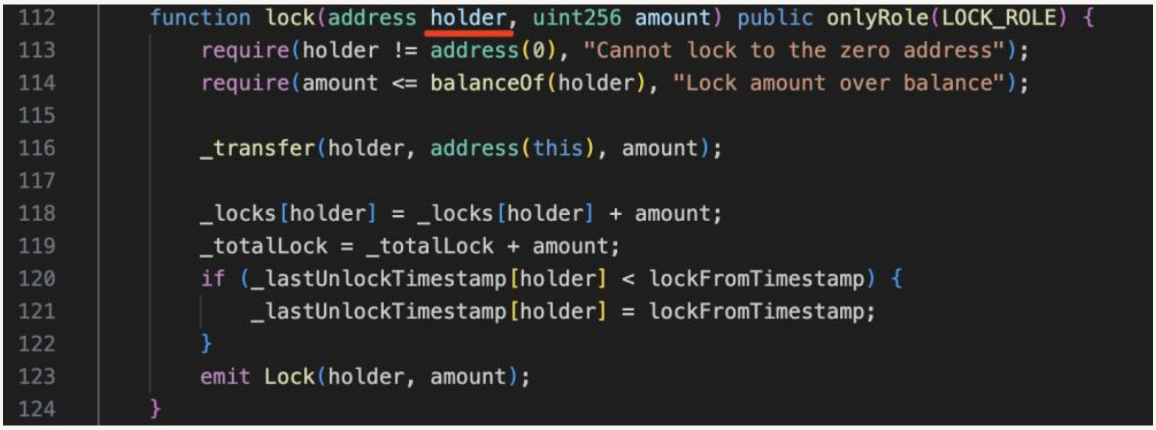
3.4 Lock token issue

ID:	NVE-004	**Location:**	Air.sol
Severity:	Medium	**Category:**	Business Issues
Likelihood:	Low	**Impact:**	Medium

Description:

The contract's LOCK_ROLE can lock any address tokens, and the permissions are too large.

When locking tokens, you should add a judgment that the holder address is not the address of this contract.



```
    function lock(address holder, uint256 amount) public onlyRole(LOCK_ROLE) {
        require(holder != address(0), "Cannot lock to the zero address");
        require(amount <= balanceOf(holder), "Lock amount over balance");

        _transfer(holder, address(this), amount);

        _locks[holder] = _locks[holder] + amount;
        _totalLock = _totalLock + amount;
        if (_lastUnlockTimestamp[holder] < lockFromTimestamp) {
            _lastUnlockTimestamp[holder] = lockFromTimestamp;
        }
        emit Lock(holder, amount);
    }
```

Figure 3.4.1 Air.sol

Recommendations:

ExVul Web3 Labs recommends the contract owner is managed using multi-signatures.

Result: Confirmed

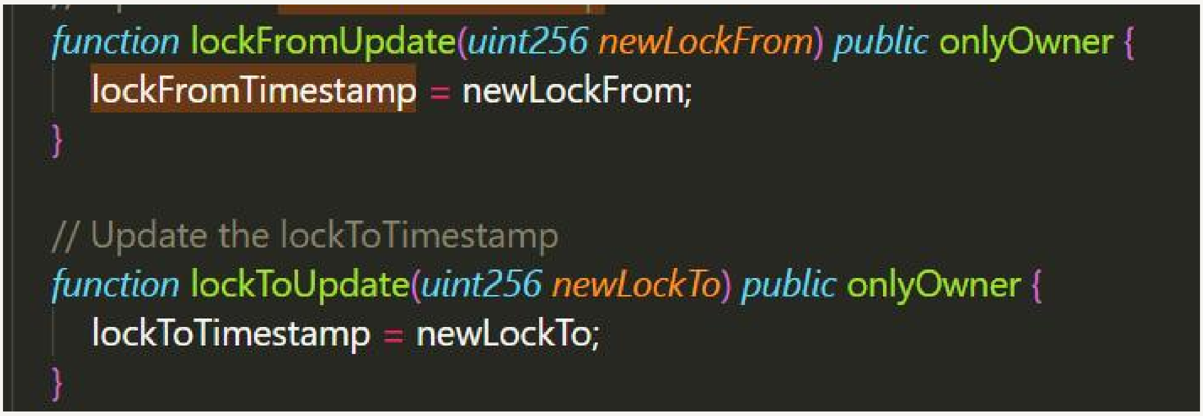
Fix Result: Ignored

3.5 Time setting logic issue

ID:	NVE-005	**Location:**	Air.sol
Severity:	Medium	**Category:**	Business Issues
Likelihood:	Medium	**Impact:**	Low

Description:

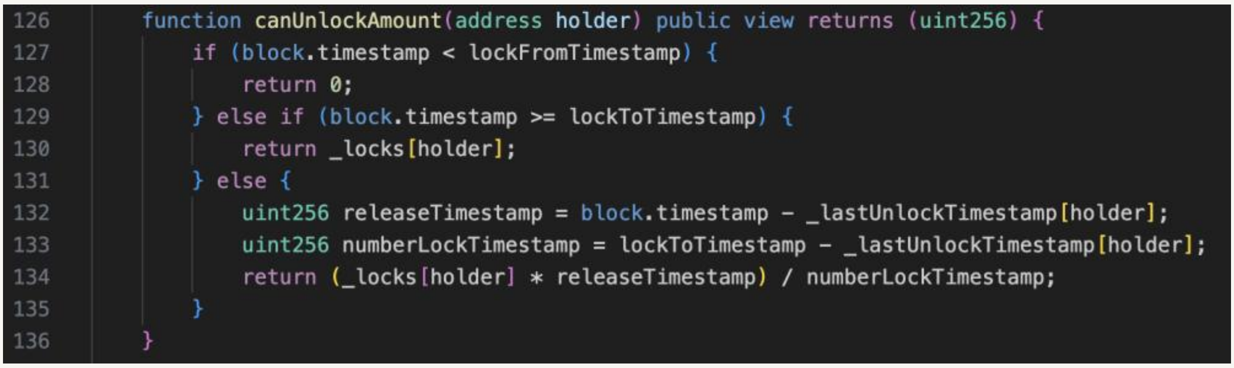
When setting the lock parameters, add the judgment that lockFromTimestamp is less than lockToTimestamp. Otherwise, the third logic cannot be run in the canUnlockAmount function, which may cause function exceptions.



```
function lockFromUpdate(uint256 newLockFrom) public onlyOwner {
    lockFromTimestamp = newLockFrom;
}

// Update the lockToTimestamp
function lockToUpdate(uint256 newLockTo) public onlyOwner {
    lockToTimestamp = newLockTo;
}
```

Figure 3.5.1 Air.sol



```
    function canUnlockAmount(address holder) public view returns (uint256) {
        if (block.timestamp < lockFromTimestamp) {
            return 0;
        } else if (block.timestamp >= lockToTimestamp) {
            return _locks[holder];
        } else {
            uint256 releaseTimestamp = block.timestamp - _lastUnlockTimestamp[holder];
            uint256 numberLockTimestamp = lockToTimestamp - _lastUnlockTimestamp[holder];
            return (_locks[holder] * releaseTimestamp) / numberLockTimestamp;
        }
    }
```

Figure 3.5.2 Air.sol

Recommendations:

ExVul Web3 Labs recommends adding the judgment that lockFromTimestamp is less than lockToTimestamp.

Result: Confirmed

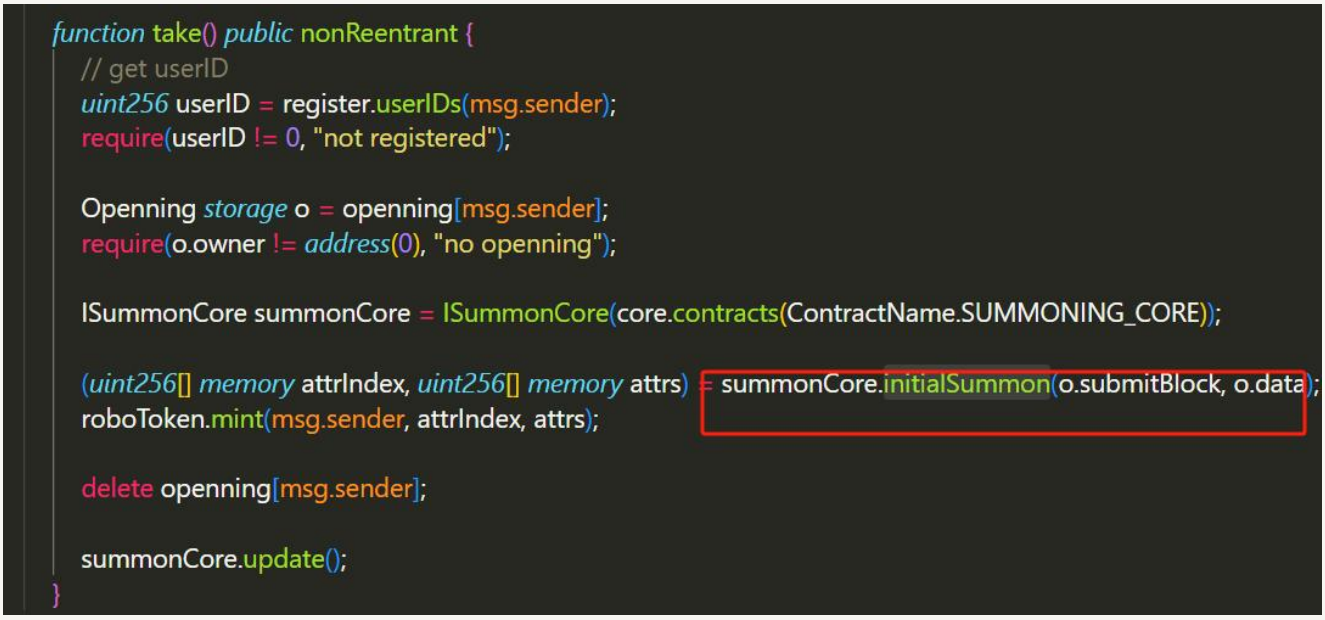
Fix Result: Ignored

3.6 Random numbers are predicted

ID:	NVE-006	**Location:**	RoBo.sol
Severity:	Medium	**Category:**	Business Issues
Likelihood:	Medium	**Impact:**	Medium

Description:

As shown in the figure below, the random number calculation in the contract may be predicted, which may affect the attribute information of NFT mint.



```
function take() public nonReentrant {
    // get userID
    uint256 userID = register.userIDs(msg.sender);
    require(userID != 0, "not registered");

    Openning storage o = openning[msg.sender];
    require(o.owner != address(0), "no openning");

    ISummonCore summonCore = ISummonCore(core.contracts(ContractName.SUMMONING_CORE));

    (uint256[] memory attrIndex, uint256[] memory attrs) = summonCore.initialSummon(o.submitBlock, o.data);
    roboToken.mint(msg.sender, attrIndex, attrs);

    delete openning[msg.sender];

    summonCore.update();
}
```

Figure 3.6.1 AirBox.sol

Recommendations:

ExVul Web3 Labs recommends to modify the code logic.

Result: Confirmed

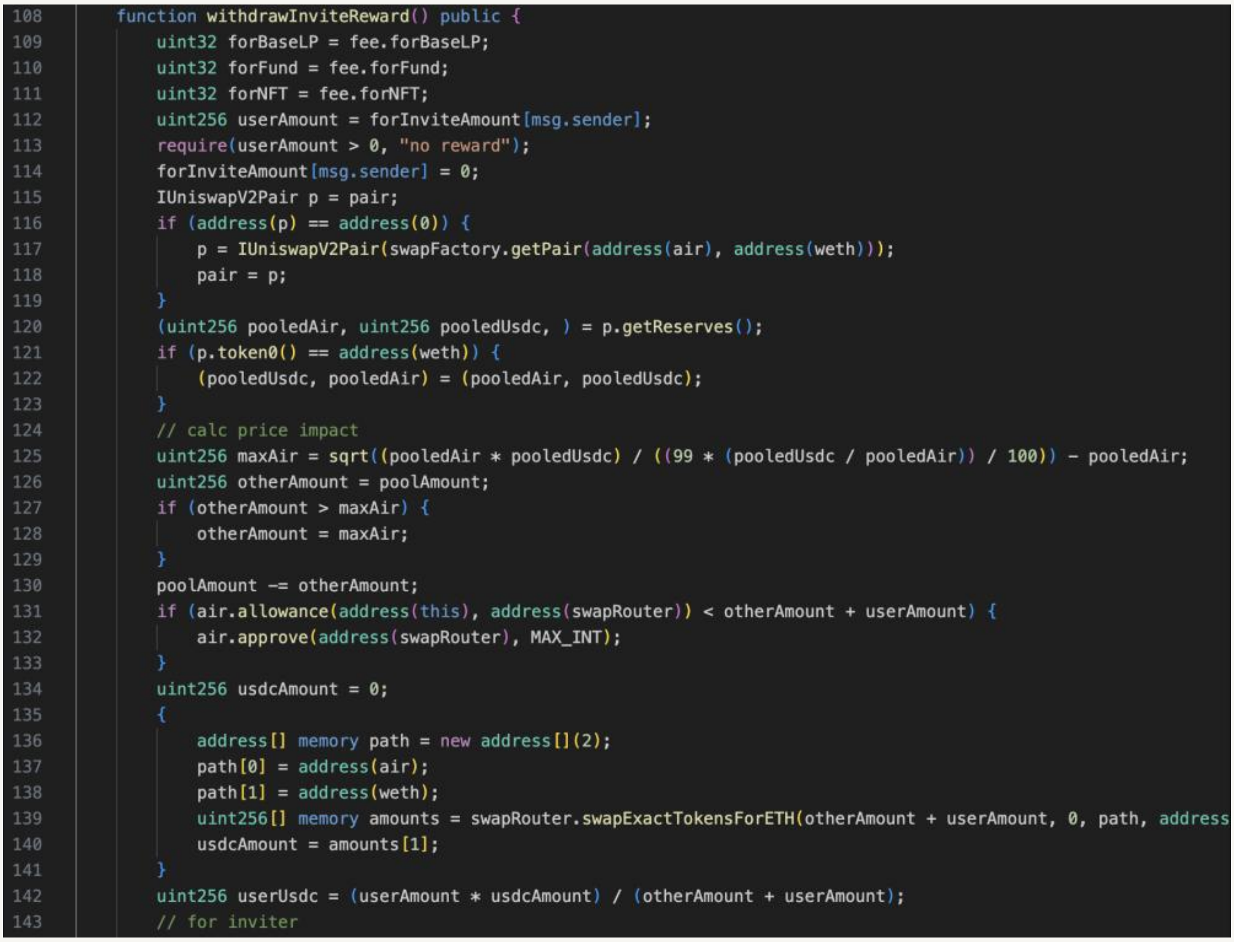
Fix Result: Ignored

3.7 withdrawInviteReward function

ID:	NVE-007	**Location:**	RoBo.sol
Severity:	Medium	**Category:**	Business Issues
Likelihood:	Medium	**Impact:**	Medium

Description:

There is uniswap's redemption function in the withdrawInviteReward function. An attack can use flash loan funds to conduct a large amount of exchanges before calling the withdrawInviteReward method, which may affect the number of trading pairs. After that, the withdrawInviteReward method is called to claim rewards, but at this time the exchange in the contract may be affected and may be attacked.



```
    function withdrawInviteReward() public {
        uint32 forBaseLP = fee.forBaseLP;
        uint32 forFund = fee.forFund;
        uint32 forNFT = fee.forNFT;
        uint256 userAmount = forInviteAmount[msg.sender];
        require(userAmount > 0, "no reward");
        forInviteAmount[msg.sender] = 0;
        IUniswapV2Pair p = pair;
        if (address(p) == address(0)) {
            p = IUniswapV2Pair(swapFactory.getPair(address(air), address(weth)));
            pair = p;
        }
        (uint256 pooledAir, uint256 pooledUsdc, ) = p.getReserves();
        if (p.token0() == address(weth)) {
            (pooledUsdc, pooledAir) = (pooledAir, pooledUsdc);
        }
        // calc price impact
        uint256 maxAir = sqrt((pooledAir * pooledUsdc) / ((99 * (pooledUsdc / pooledAir)) / 100)) - pooledAir;
        uint256 otherAmount = poolAmount;
        if (otherAmount > maxAir) {
            otherAmount = maxAir;
        }
        poolAmount -= otherAmount;
        if (air.allowance(address(this), address(swapRouter)) < otherAmount + userAmount) {
            air.approve(address(swapRouter), MAX_INT);
        }
        uint256 usdcAmount = 0;
        {
            address[] memory path = new address[](2);
            path[0] = address(air);
            path[1] = address(weth);
            uint256[] memory amounts = swapRouter.swapExactTokensForETH(otherAmount + userAmount, 0, path, address
            usdcAmount = amounts[1];
        }
        uint256 userUsdc = (userAmount * usdcAmount) / (otherAmount + userAmount);
        // for inviter
```

Figure 3.7.1 AirBox.sol

Recommendations:

ExVul Web3 Labs recommends to modify the code logic.

Result: Confirmed

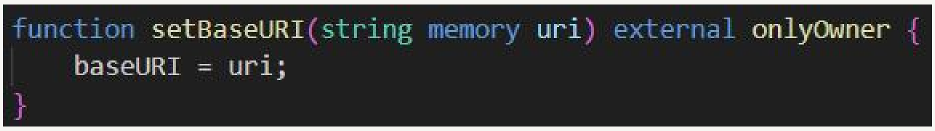
Fix Result: Ignored

3.8 NFT URI issue

ID:	NVE-008	**Location:**	RoBo.sol
Severity:	Low	**Category:**	Business Issues
Likelihood:	Low	**Impact:**	Low

Description:

As shown in the figure below, The contract owner can modify the baseURI of the NFT. After modified, it will cause the NFT query link to change.



```
function setBaseURI(string memory uri) external onlyOwner {
    baseURI = uri;
}
```

Figure 3.8.1 RoBo.sol

Recommendations:

ExVul Web3 Labs recommends the contract owner is managed using multi-signatures.

Result: Confirmed

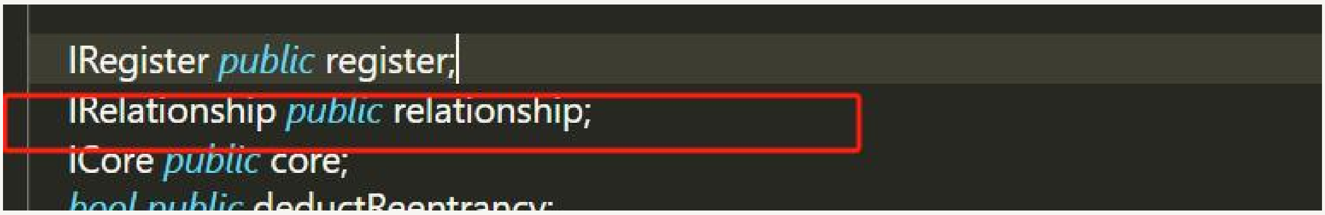
Fix Result: Ignored

3.9 Redundant code

ID:	NVE-009	**Location:**	Air.sol
Severity:	Informational	**Category:**	Business Issues
Likelihood:	Informational	**Impact:**	Informational

Description:

The IRelationship interface are not used.



```
IRegister public register;
IRelationship public relationship;
ICore public core;
bool public deductReentrancy;
```

Figure 3 .9.1 Air.sol

Recommendations:

ExVul Web3 Labs recommends removing unused interface .

Result: Confirmed

Fix Result: Ignored

4. CONCLUSION

In this audit, we thoroughly analyzed **Meta** smart contract implementation. The problems found are described and explained in detail in Section 3. The problems found in the audit have been communicated to the project leader. We therefore consider the audit result to be **PASSED**. To improve this report, we greatly appreciate any constructive feedbacks or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

5. APPENDIX

5.1 Basic Coding Assessment

5.1.1 Apply Verification Control

- Description: The security of apply verification
- Result: Not found
- Severity: Critical

5.1.2 Authorization Access Control

- Description: Permission checks for external integral functions
- Result: Not found
- Severity: Critical

5.1.3 Forged Transfer Vulnerability

- Description: Assess whether there is a forged transfer notification vulnerability in the contract
- Result: Not found
- Severity: Critical

5.1.4 Transaction Rollback Attack

- Description: Assess whether there is transaction rollback attack vulnerability in the contract.
- Result: Not found
- Severity: Critical

5.1.5 Transaction Block Stuffing Attack

- Description: Assess whether there is transaction blocking attack vulnerability.
- Result: Not found
- Severity: Critical

5.1.6 Soft Fail Attack Assessment

- Description: Assess whether there is soft fail attack vulnerability.
- Result: Not found
- Severity: Critical

5.1.7 Hard Fail Attack Assessment

- Description: Examine for hard fail attack vulnerability
- Result: Not found
- Severity: Critical

5.1.8 Abnormal Memo Assessment

- Description: Assess whether there is abnormal memo vulnerability in the contract.

- Result: Not found
- Severity: Critical

5.1.9 Abnormal Resource Consumption

- Description: Examine whether abnormal resource consumption in contract processing.
- Result: Not found
- Severity: Critical

5.1.10 Random Number Security

- Description: Examine whether the code uses insecure random number.
- Result: Not found
- Severity: Critical

5.2 Advanced Code Scrutiny

5.2.1 Cryptography Security

- Description: Examine for weakness in cryptograph implementation.
- Results: Not Found
- Severity: High

5.2.2 Account Permission Control

- Description: Examine permission control issue in the contract
- Results: Not Found
- Severity: Medium

5.2.3 Malicious Code Behavior

- Description: Examine whether sensitive behavior present in the code
- Results: Not found
- Severity: Medium

5.2.4 Sensitive Information Disclosure

- Description: Examine whether sensitive information disclosure issue present in the code.
- Result: Not found
- Severity: Medium

5.2.5 System API

- Description: Examine whether system API application issue present in the code
- Results: Not found
- Severity: Low

6. DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without ExVul's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts ExVul to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. ExVul's position is that each company and individual are responsible for their own due diligence and continuous security. ExVul's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

7. REFERENCES

[1] MITRE. CWE- 191: Integer Underflow (Wrap or Wraparound).

https://cwe.mitre.org/data/ definitions/191.html.

[2] MITRE. CWE- 197: Numeric Truncation Error.

https://cwe.mitre.org/data/definitions/197. html.

[3] MITRE. CWE-400: Uncontrolled Resource Consumption.

https://cwe.mitre.org/data/ definitions/400.html.

[4] MITRE. CWE-440: Expected Behavior Violation.

https://cwe.mitre.org/data/definitions/440. html.

[5] MITRE. CWE-684: Protection Mechanism Failure.

https://cwe.mitre.org/data/definitions/ 693.html.

[6] MITRE. CWE CATEGORY: 7PK - Security Features.

https://cwe.mitre.org/data/definitions/ 254.html.

[7] MITRE. CWE CATEGORY: Behavioral Problems.

https://cwe.mitre.org/data/definitions/438. html.

[8] MITRE. CWE CATEGORY: Numeric Errors.

https://cwe.mitre.org/data/definitions/189.html.

[9] MITRE. CWE CATEGORY: Resource Management Errors.

https://cwe.mitre.org/data/ definitions/399.html.

[10] OWASP. Risk Rating Methodology.

https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

www.exvul.com

contact@exvul.com

@exvul

github.com/ExVul

ExVul