Smart Contract Audit Report

LuckyNFT Smart Contract

23 Dec 2022

Numen Cyber Labs - Security Services

Table of Content

1 Executive Summary

Numen Cyber Technology was engaged by LuckyNFT 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.

In our first audit, One Critical, One High, One Medium and Three Informational severities findings are related to owner authority, centralized risk, out of gas and unfair lottery.

After modifying by developers with our proposal, one Informational severities findings related to owner authority was leaved. And the developers chose to ignore that issue and will deployed the contract with multisigWallet and timelock.

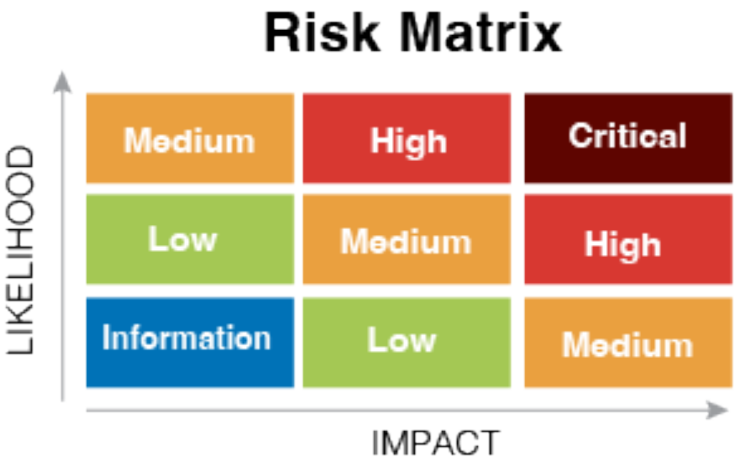
The outcome of the assessment outlined in chapter 3 provides the system's owners a full description of the vulnerabilities identified, the associated risk rating for each vulnerability, and detailed recommendations that will resolve the underlying technical issue.

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 and impact are categorized into three ratings: High, Medium and Low. Severity is determined by likelihood and impact and can be classified into four categories accordingly, Critical, High, Medium, Low shown in table 1.1.



Risk Matrix

LIKELIHOOD			
	Medium	High	Critical
	Low	Medium	High
	Information	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	Asset Security

Code Scrutiny	Cryptography Security
	Business Logic Review
	Source Code Functional Verification
	Account Authorization Control
	Sensitive Information Disclosure
	Circuit Breaker
	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: LuckyNFT

Project URL: https://oubn.test.chainser.cn/

Audit Time: 2022/12.20 - 2022/12.26

Language: solidity

Contract Name	Smart Contract Hash (sha256)
ERC721GWhitelistAuth/ERC721G.sol	4731a36780db01e41a55f35e147a1b1c5fe0c1ad1071d4a2969d3153619d2001
ERC721GWhitelistAuth/Lucky.sol	08b3228af55250dc49dbd6c05daa6c8ac132e1e474d99c09da58ed5d6c4fc4af
LuckyNFT.sol	322e5899e9b467d3cd274ba2bd4a4a180358446e9e0d0abbbd0709d0f5fc4b98
LuckyNFTPool.sol	ae6324cce8df2eb377aa253ac1629026d72c80a052c15438174bc785ff9d1912
LuckyNFTRakeBack.sol	9fe957fa82314a4870f9f15a092657564f36e045205e6c5748ef69a51d6dc638
OpenseaExchangeProxy.sol	f8d9f90a58b51d7fa8fb9259f7762c3a245d127c7c826e35c77f4880cac5d964
OpenseaSeaportProxy.sol	f3b5c33ed7c10969d09cd8b47a4aea23de96c78b27ffdcdc4ca0e33eecaa9980
Utils.sol	5b2566393f445d502eea461a8960f6e0d75bd14dc230242b8c8f7e27852a31e7

2.2 Summary

Severity	Found	
Critical	1	⬛
High	1	🟥
Medium	1	🟨
Low	0	
Informational	3	⬜ ⬜ ⬜

2.3 Key Findings

Two Medium severities findings are related to owner authority, centralized risk.

ID	Severity	Findings Title	Status	Confirm
NVE-001	Critical	Owner can change oracle	fixed	true
NVE-002	Informational	Owner can change trade proxy	Ignored	true
NVE-003	Medium	Out of gas	fixed	true
NVE-004	High	Unfair lottery	fixed	true
NVE-005	Informational	Unnecessary library imported	fixed	true

NVE-006	Informational	Unnecessary determine statements	fixed	true

Table 2.1: Key Audit Findings

3 Detailed Description of Findings

~~3.1 Owner can change oracle~~

ID: NVE-001

Location: LuckyNFT.sol

Severity: Critical

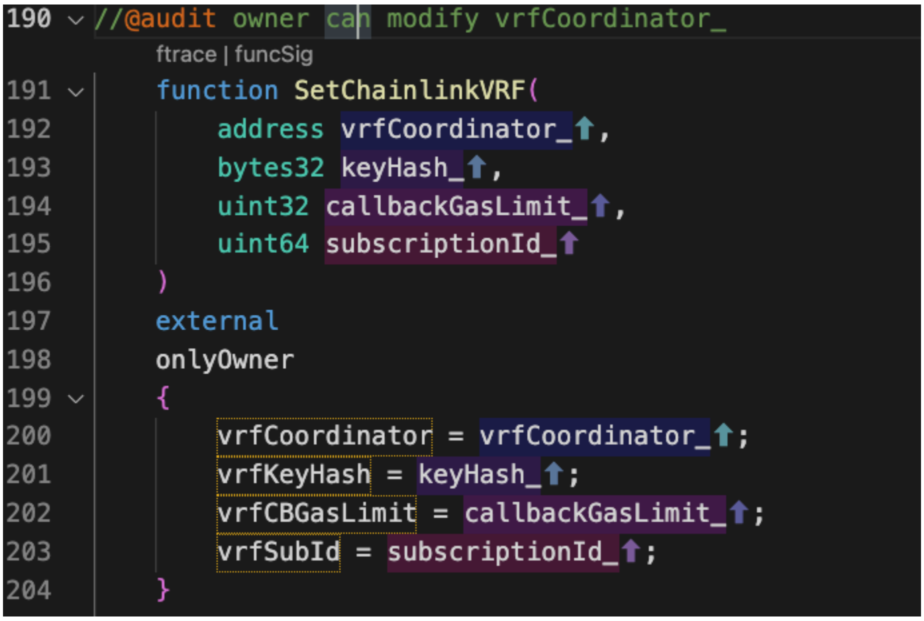
Category: Authority Issues

Likelihood: Critical

Impact: Critical

Description:

The LuckyNFT contract is a loot box contract that crowdfund a NFT but just one share can be winner and get the NFT. The Random words are request from Chainlink oracle, and use the version 0.8 VRF. However, the owner of this contract can change the address of vrfCoordinator in function SetChainlinkVRF. If vrfCoordinator is changed, it will have the risk for a designed random words and loose the fairness of the business. The specific code segment is shown in the Figure 1.



```
//@audit owner can modify vrfCoordinator_
    function SetChainlinkVRF(
        address vrfCoordinator_,
        bytes32 keyHash_,
        uint32 callbackGasLimit_,
        uint64 subscriptionId_
    )
    external
    onlyOwner
    {
        vrfCoordinator = vrfCoordinator_;
        vrfKeyHash = keyHash_;
        vrfCBGasLimit = callbackGasLimit_;
        vrfSubId = subscriptionId_;
    }
```

Figure 1 function SetChainlinkVRF

Recommendations:

Numen Cyber Lab recommends proper management of private keys or use Gnosis multisig for owner address. And the vrfCoordinator should be immutable.

Result: Critical

Fix Result: Fixed, The Hash value of the latest version LuckyNFT.sol is

8b1be5df81d6785978d5d66d4d951f681f5a2259f2fffaf1ad1a40dc693b951d

3.2 OWNER CAN CHANGE TRADEPROXY

ID: NVE-002 Location: LuckyNFT.sol

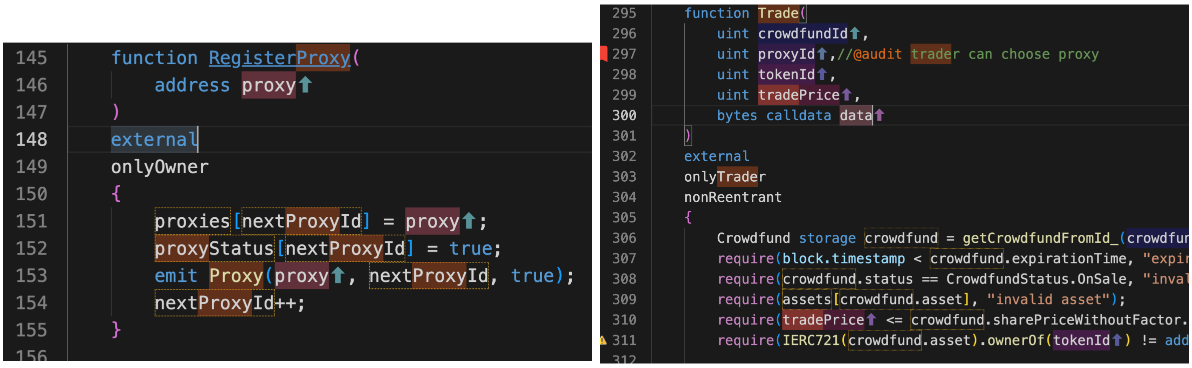
Severity: Informational Category: Authority Issues

Likelihood: Informational

Impact: Informational

Description:

The LuckyNFT contract's crowd fund trader can call the function trade to buy NFT from opensea with the 2 opensea trading proxy contracts and trigger the loot box opened. However, the owner of this contract can register multiple addresses of trading proxy contracts and the trader will not trade from opensea by choose a registered proxy. The specific code segment is shown in the Figure 2.



```
    function RegisterProxy(
        address proxy
    )
    external
    onlyOwner
    {
        proxies[nextProxyId] = proxy;
        proxyStatus[nextProxyId] = true;
        emit Proxy(proxy, nextProxyId, true);
        nextProxyId++;
    }
```

```
    function Trade(
        uint crowdfundId,
        uint proxyId,//@audit trader can choose proxy
        uint tokenId,
        uint tradePrice,
        bytes calldata data
    )
    external
    onlyTrader
    nonReentrant
    {
        Crowdfund storage crowdfund = getCrowdfundFromId_(crowdfun
        require(block.timestamp < crowdfund.expirationTime, "expir
        require(crowdfund.status == CrowdfundStatus.OnSale, "inval
        require(assets[crowdfund.asset], "invalid asset");
        require(tradePrice <= crowdfund.sharePriceWithoutFactor.
        require(IERC721(crowdfund.asset).ownerOf(tokenId) != add
```

Figure 2 function RegisterProxy & Trade

Recommendations:

Numen Cyber Lab recommends the openseaTarget address in opensea proxy contracts should be verified.

Result: Informational

Fix Result: Ignored. The developer will deployed the owner with multisigWallet and timelock.

The Hash value of the two contracts is

MultiSigWallet/MultiSigWallet.sol	0048fe7c4636bbc2a0afc9b1997add25e0338393fcf25da3e94459969bf06052
MultiSigWallet/MultiSigWalletWithTimelock.sol	d1f0b32f25fcde5fa7d2d2af523b29b1d5c4865255bd145682d207858e881aa1

~~3.3 OUT OF GAS~~

ID: NVE-003 Location: LuckyNFT.sol, Lucky.sol

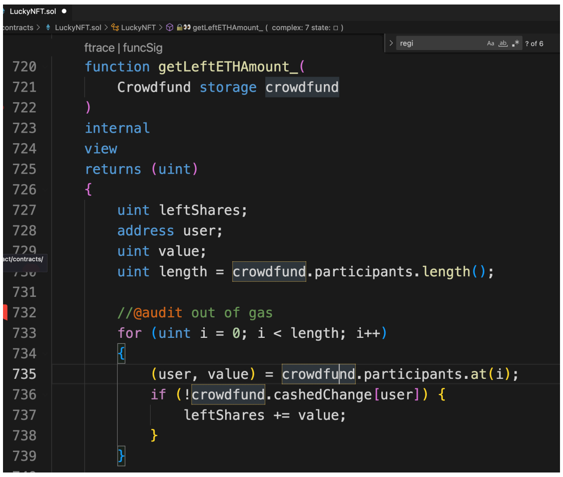
Severity: Medium Category: Basic Coding Assessment

Likelihood: Medium

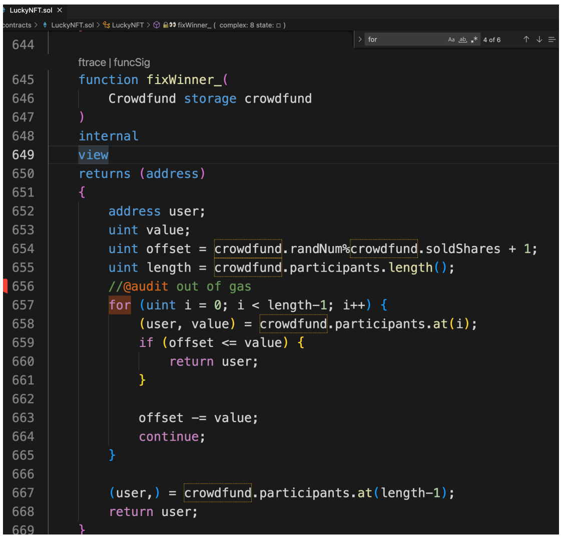
Impact: Medium

Description:

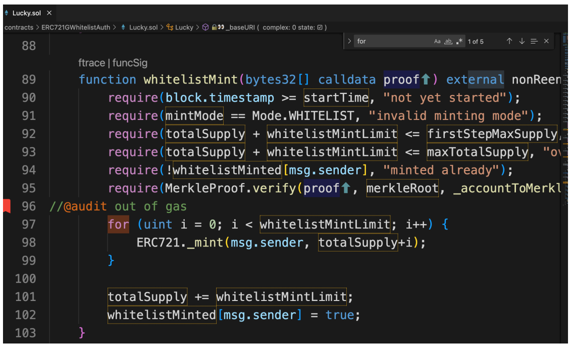
In the LuckyNFT & Lucky contracts, there are some for loops may become big loop and cause the out of gas issue. The specific code segment is shown in the Figure 3.


LuckyNFT.sol
contracts > LuckyNFT.sol > LuckyNFT > getLeftETHAmount_ (complex: 7 state: □)
regi
? of 6
act/contracts/

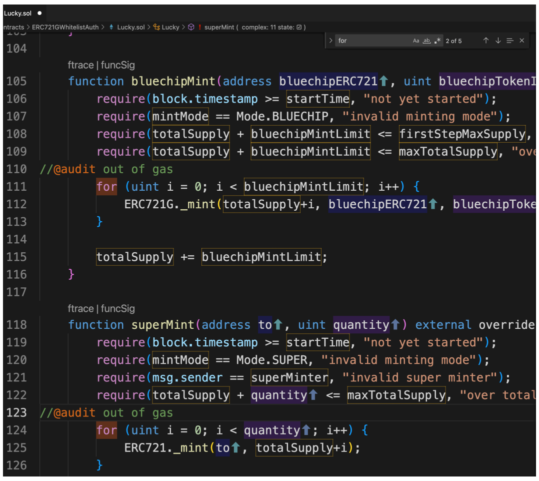
```
ftrace | funcSig
function getLeftETHAmount_(
    Crowdfund storage crowdfund
)
internal
view
returns (uint)
{
    uint leftShares;
    address user;
    uint value;
    uint length = crowdfund.participants.length();

    //@audit out of gas
    for (uint i = 0; i < length; i++)
    {
        (user, value) = crowdfund.participants.at(i);
        if (!crowdfund.cashedChange[user]) {
            leftShares += value;
        }
    }
```


LuckyNFT.sol
contracts > LuckyNFT.sol > LuckyNFT > fixWinner_ (complex: 8 state: □)
for
4 of 6

```

ftrace | funcSig
function fixWinner_(
    Crowdfund storage crowdfund
)
internal
view
returns (address)
{
    address user;
    uint value;
    uint offset = crowdfund.randNum%crowdfund.soldShares + 1;
    uint length = crowdfund.participants.length();
    //@audit out of gas
    for (uint i = 0; i < length-1; i++) {
        (user, value) = crowdfund.participants.at(i);
        if (offset <= value) {
            return user;
        }

        offset -= value;
        continue;
    }

    (user,) = crowdfund.participants.at(length-1);
    return user;
}
```


Lucky.sol
contracts > ERC721GWhitelistAuth > Lucky.sol > Lucky > _baseURI (complex: 0 state: ☑)
for
1 of 5

```

ftrace | funcSig
function whitelistMint(bytes32[] calldata proof) external nonReen
    require(block.timestamp >= startTime, "not yet started");
    require(mintMode == Mode.WHITELIST, "invalid minting mode");
    require(totalSupply + whitelistMintLimit <= firstStepMaxSupply,
    require(totalSupply + whitelistMintLimit <= maxTotalSupply, "ov
    require(!whitelistMinted[msg.sender], "minted already");
    require(MerkleProof.verify(proof, merkleRoot, _accountToMerkl
//@audit out of gas
    for (uint i = 0; i < whitelistMintLimit; i++) {
        ERC721._mint(msg.sender, totalSupply+i);
    }

    totalSupply += whitelistMintLimit;
    whitelistMinted[msg.sender] = true;
}
```


Lucky.sol
ntracts > ERC721GWhitelistAuth > Lucky.sol > Lucky > ! superMint (complex: 11 state: ☑)
for
2 of 5

```

ftrace | funcSig
function bluechipMint(address bluechipERC721, uint bluechipTokenI
    require(block.timestamp >= startTime, "not yet started");
    require(mintMode == Mode.BLUECHIP, "invalid minting mode");
    require(totalSupply + bluechipMintLimit <= firstStepMaxSupply,
    require(totalSupply + bluechipMintLimit <= maxTotalSupply, "ove
//@audit out of gas
    for (uint i = 0; i < bluechipMintLimit; i++) {
        ERC721G._mint(totalSupply+i, bluechipERC721, bluechipToke
    }

    totalSupply += bluechipMintLimit;
}

ftrace | funcSig
function superMint(address to, uint quantity) external override
    require(block.timestamp >= startTime, "not yet started");
    require(mintMode == Mode.SUPER, "invalid minting mode");
    require(msg.sender == superMinter, "invalid super minter");
    require(totalSupply + quantity <= maxTotalSupply, "over total
//@audit out of gas
    for (uint i = 0; i < quantity; i++) {
        ERC721._mint(to, totalSupply+i);
    }
```

Figure 3 out of gas functions

Recommendations:

Numen Cyber Lab recommends to limit the loop times.

Result: Medium

Fix Result: Fixed. the Hash value of the latest version LuckyNFT.sol is

8b1be5df81d6785978d5d66d4d951f681f5a2259f2fffaf1ad1a40dc693b951d

~~3.4 Unfair lottery~~

ID: NVE-004

Location: LuckyNFT.sol

Severity: High

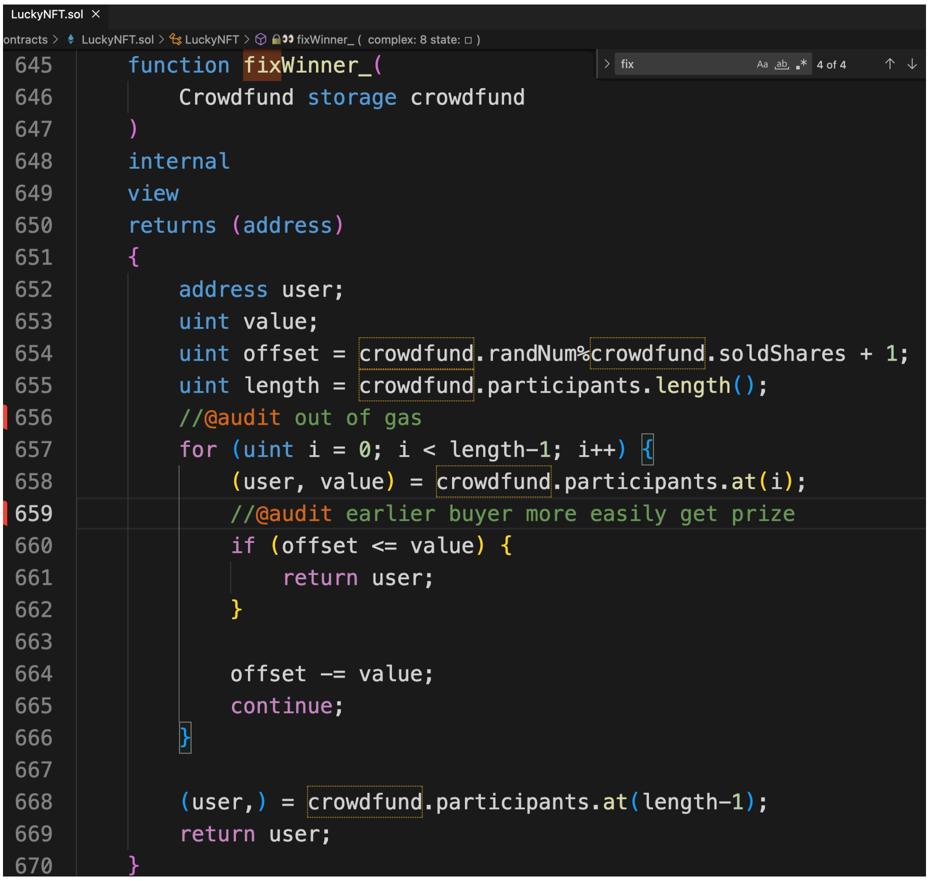
Category: Basic Coding Assessment

Likelihood: High

Impact: High

Description:

In the LuckyNFT contract, the lucky user will be decided in fixWinner_ function after the chainlink calling back. However, the logic in fixWinner_ will make an unfair lottery result, because the earlier shares buyer can get prize more easily. The specific code segment is shown in the Figure 4.


LuckyNFT.sol ×
ontracts > LuckyNFT.sol > LuckyNFT > fixWinner_ (complex: 8 state: □)
fix
Aa
ab
.*
4 of 4

```
    function fixWinner_(
        Crowdfund storage crowdfund
    )
    internal
    view
    returns (address)
    {
        address user;
        uint value;
        uint offset = crowdfund.randNum%crowdfund.soldShares + 1;
        uint length = crowdfund.participants.length();
        //@audit out of gas
        for (uint i = 0; i < length-1; i++) {
            (user, value) = crowdfund.participants.at(i);
            //@audit earlier buyer more easily get prize
            if (offset <= value) {
                return user;
            }

            offset -= value;
            continue;
        }

        (user,) = crowdfund.participants.at(length-1);
        return user;
    }
```

*Figure 4 function fixWinner_*

Recommendations:

Numen Cyber Lab recommends to use equal probability mechanism.

Result: High

Fix Result: Fixed, The Hash value of the latest version LuckyNFT.sol is

8b1be5df81d6785978d5d66d4d951f681f5a2259f2fffaf1ad1a40dc693b951d

~~3.5 Unnecessary library imported~~

ID: NVE-005 Location: LuckyNFT.sol

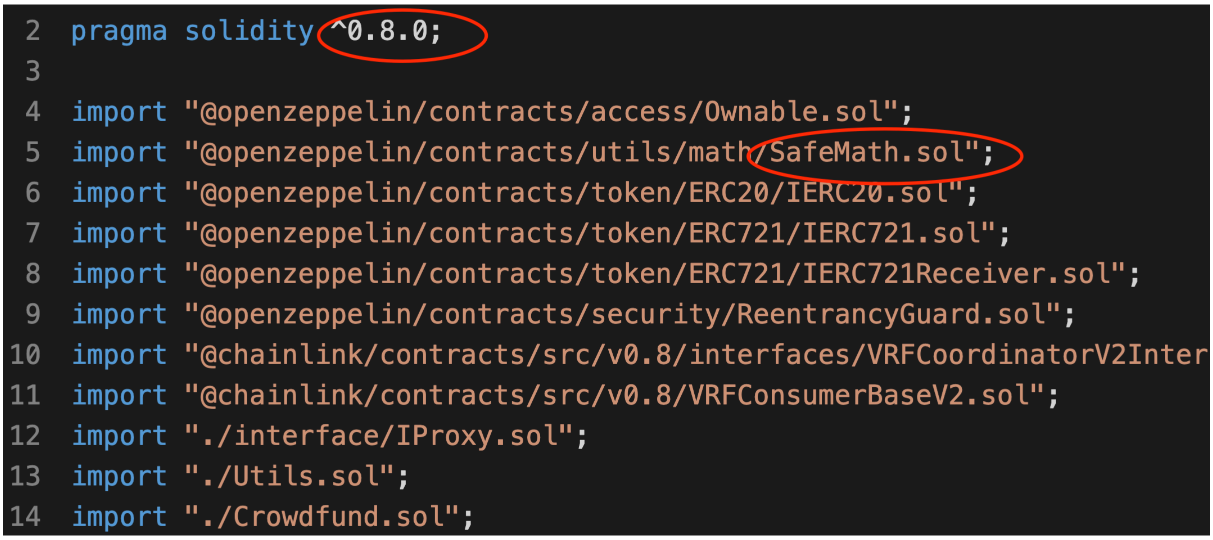
Severity: Informational Category: Basic Coding Assessment

Likelihood: Informational

Impact: Informational

Description:

In solidity v0.8, if not use uncheck key word, it is not necessary to import SafeMath library. The specific code segment is shown in the Figure 5.



```
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/math/SafeMath.sol";
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/token/ERC721/IERC721.sol";
import "@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol";
import "@openzeppelin/contracts/security/ReentrancyGuard.sol";
import "@chainlink/contracts/src/v0.8/interfaces/VRFCoordinatorV2Inter
import "@chainlink/contracts/src/v0.8/VRFConsumerBaseV2.sol";
import "./interface/IProxy.sol";
import "./Utils.sol";
import "./Crowdfund.sol";
```

Figure 5 no need to use safemath

Recommendations:

Numen Cyber Lab recommends to not use safemath and tryxxx methods.

Result: Informational

Fix Result: Fixed, The Hash value of the latest version LuckyNFT.sol is

8b1be5df81d6785978d5d66d4d951f681f5a2259f2fffaf1ad1a40dc693b951d

~~3.6 UNNECESSARY DETERMINE STATEMENT~~

ID: NVE-006

Location: LuckyNFT.sol

Severity: Informational

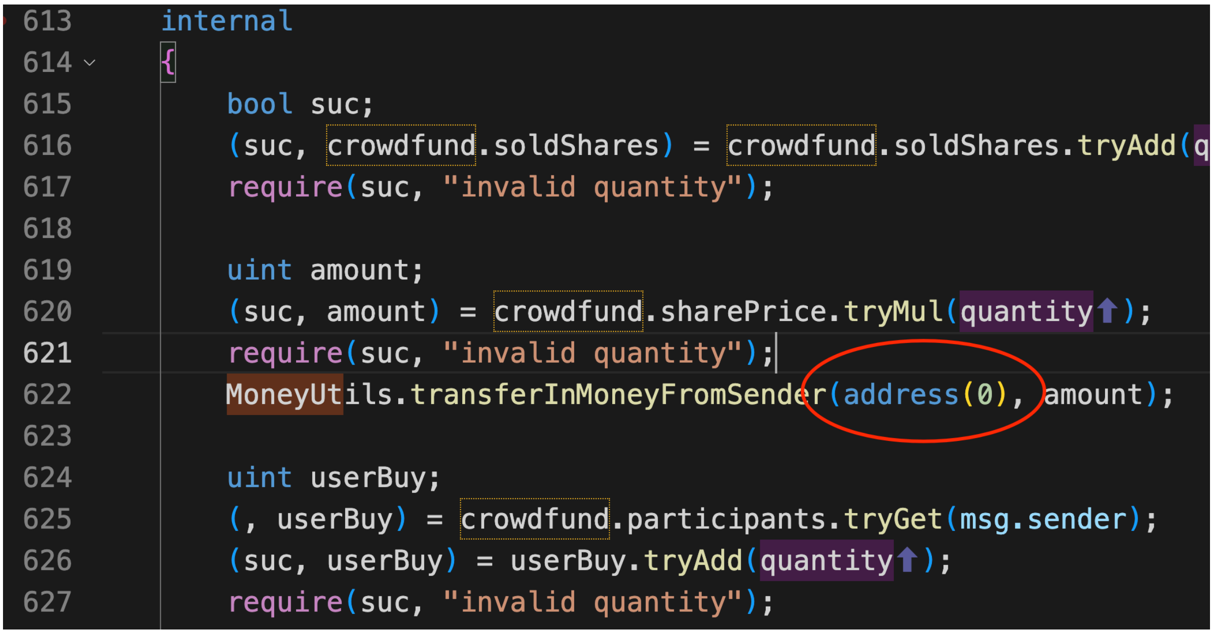
Category: Basic Coding Assessment

Likelihood: Informational

Impact: Informational

Description:

Because of all token asset is native coin, it is not necessary to use the MoneyUtils to add determine statements. The specific code segment is shown in the Figure 6.



```
    internal
    {
        bool suc;
        (suc, crowdfund.soldShares) = crowdfund.soldShares.tryAdd(q
        require(suc, "invalid quantity");

        uint amount;
        (suc, amount) = crowdfund.sharePrice.tryMul(quantity);
        require(suc, "invalid quantity");
        MoneyUtils.transferInMoneyFromSender(address(0), amount);

        uint userBuy;
        (, userBuy) = crowdfund.participants.tryGet(msg.sender);
        (suc, userBuy) = userBuy.tryAdd(quantity);
        require(suc, "invalid quantity");
```

Figure 6 no need to use MoneyUtils

Recommendations:

Numen Cyber Lab recommends to use transfer or call.value() directly.

Result: Informational

Fix Result: Fixed, The Hash value of the latest version LuckyNFT.sol is

8b1be5df81d6785978d5d66d4d951f681f5a2259f2fffaf1ad1a40dc693b951d

4 CONCLUSION

In this audit, we thoroughly analysed LuckyNFT smart contract implementation. The problems found are described and explained in detail in Section 3. The problems found in the audit have been brought up to the project party, ignored issues are in line with the project design, and the contracts will be deployed with multisigWallet and timelock. We therefore deem the audit result to be a **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: 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 Behaviour

- Description: Examine whether sensitive behaviour 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 Numen'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 Numen 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. Numen's position is that each company and individual are responsible for their own due diligence and continuous security. Numen'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.

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

Numen Cyber Technology Pte. Ltd.

11 North Buona Vista Drive, #04-09,

The Metropolis, Singapore 138589

Tel: 65-63555555

Fax: 65-63666666

Email: sales@numencyber.com

Web: https://numencyber.com