Making Web3 Space Safer for Everyone

KALOS

kingdomNFT V1.0

Security Assessment

Published on : 11 Oct. 2023
Version v1.1

Security Report Published by KALOS

v1.1 11 Oct. 2023

Auditor : Jinu Lee

Found issues

Severity of Issues	Findings	Resolved	Acknowledged	Comment
Critical	1	1	-	-
High	4	4	-	-
Medium	2	2	-	-
Low	3	3	-	-
Tips	5	4	1	-

TABLE OF CONTENTS

TABLE OF CONTENTS
ABOUT US
Executive Summary
OVERVIEW
- Contract Overview
- Scope
- Access Controls

FINDINGS
- 1. KingdomStoryMarket design issue
- 2. Insufficient data in sign messages leads to potential signature issues
- 3. KingdomStoryMysteryBox unsafe random value
- 4. UnlockAndBurn function without owner authentication implemented
- 5. KingdomStoryHeroNFT contract reentrancy
- 6. KingdomStoryHeroNFT signature reuse issue 1
- 7. KingdomStoryHeroNFT signature reuse issue 2
- 8. KingdomStoryHeroNFT-UnlockAndSetTokenURI insufficient tokenId validation
- 9. Upgradeable issue
- 10. function/variables naming style
- 11. unused functions
- 12. non-implemented pausable functions
- 13. Missing event logging in several functions
- 14. Incorrect implementation due to mixing of token value and native-coin value in multiTransferEther function of MultiTransferEtherToken
- 15. KingdomStoryMysteryBox contract reentrancy

DISCLAIMER
Appendix. A
- Severity Level
- Difficulty Level
- Vulnerability Category

ABOUT US

Making Web3 Space Safer for Everyone

KALOS is a flagship service of HAECHI LABS, the leader of the global blockchain industry. We bring together the best Web2 and Web3 experts. Security Researchers with expertise in cryptography, leaders of the global best hacker team, and blockchain/smart contract experts are responsible for securing your Web3 service.

Having secured $60B crypto assets on over 400 main-nets, Defi protocols, NFT services, P2E, and Bridges, KALOS is the only blockchain technology company selected for the Samsung Electronics Startup Incubation Program in recognition of our expertise. We have also received technology grants from the Ethereum Foundation and Ethereum Community Fund.

Inquiries: audit@kalos.xyz
Website: https://kalos.xyz

Executive Summary

Purpose of this report

This report was prepared to audit the security of the project developed by Softnyx. KALOS conducted the audit focusing on whether the system created by the Softnyx is soundly implemented and designed as specified in the published materials, in addition to the safety and security of the project.

In detail, we have focused on the following

- Proper handling of trade in Market contract
- Proper handling of signatures in HeroNFT, Market contracts
- Proper handling of random value in MysteryBox contract
- Denial of Service
- Manipulation of important variables of owner and master permissions due to incorrect access control settings
- Existence of other known Smart Contract vulnerabilities

Codebase Submitted for the Audit

The codes used in this Audit can be found on GitHub
https://github.com/softnyx/kingdomNFT.solidity

The commit hash of the code used for this Audit is
“f120792a94e0d4bf1eeda8f261b99c2838822197”.

The commit hash of the code used for v1.1 patch review is
“713ce0dc5d661a1c2ac6aa4a858aacbaa062c02c”.

Audit Timeline

Date	Event
2023/09/06	Audit Initiation
2023/09/19	Delivery of v1.0 report.
2023/10/11	Delivery of v1.1 report.

Findings

KALOS found 1 Critical, 4 High, 2 Medium, 3 Low severity issues. There are 5 Tips issues explained that would improve the code's usability or efficiency upon modification.

Severity	Issue	Status
High	KingdomStoryMarket design issue	(Fixed - v1.1)
Low	Insufficient data in sign messages leads to potential signature issues	(Fixed - v1.0)
Low	KingdomStoryMysteryBox unsafe random value	(Fixed - v1.1)
High	UnlockAndBurn function without owner authentication implemented	(Fixed - v1.0)
High	KingdomStoryHeroNFT contract reentrancy	(Fixed - v1.0)
Medium	KingdomStoryHeroNFT signature reuse issue 1	(Fixed - v1.0)
Medium	KingdomStoryHeroNFT signature reuse issue 2	(Fixed - v1.0)
Low	KingdomStoryHeroNFT-UnlockAndSetTokenURI insufficient tokenId validation	(Fixed - v1.0)
Tips	Upgradeable issue	(Fixed - v1.0)
Tips	function/variables naming style	(Acknowledged - v1.1)
Tips	unused functions	(Fixed - v1.0)
Tips	non-implemented pausable functions	(Fixed - v1.0)

Tips	Missing event logging in several functions	(Fixed - v1.1)
Critical	Incorrect implementation due to mixing of token value and native-coin value in multiTransferEther function of MultiTransferEtherToken	(Fixed - v1.0)
High	KingdomStoryMysteryBox contract reentrancy	(Fixed - v1.0)

Remarks

KingdomStory contracts use the proxy pattern, which means that the logic of these contracts can be changed arbitrarily by Softnyx. These changes have the potential to impact the assets of users interacting with these smart contracts.

OVERVIEW

Contract Overview

KingdomStoryBKT

This contract is the ERC20 token that will be used during the beta period. In the initialize function, 300_000_000 tokens are minted to msg.sender, and the contract has a mint function, so master and owner can mint additional tokens. To prevent it from being used outside the market contract, a feature is implemented in the _beforeTokenTransfer function to verify that the from and to addresses are allowed destinations.

KingdomStoryKID

This contract is the ERC20 token that will be used during the live period. In the initialize function, 300_000_000 tokens are minted to msg.sender, and the contract has a mint function, so master and owner can mint additional tokens.

KingdomStoryHeroNFT

This contract is the ERC721 NFT where the hero's URI information is stored. This contract has a lock-nft/unlock-nft function, and NFTs that are locked can’t be transferred. This contract has a mint function, so the box contract, master and owner can mint additional nfts. If someone gets a master's signature from web2, they can change the URI information of the NFT or mint a new NFT.

KingdomStoryMarket

This contract is the NFT market contract, and the trade function between NFT and erc20-token/native-coin is implemented. When the seller signs the sales data(on web2 side), the buyer sends the sales data and signature to the market contract to purchase. Only hero/box NFT can be traded, and erc20-token has no restrictions. The master/owner can set a trading fee, and can collect a percent of the tokens used in the trade as a fee.

KingdomStoryMysteryBox

This contract is the ERC721 NFT. It is used for randomly minting KingdomStoryHeroNFTs. The NFTs in this contract are called BOX, and the box types are gold, silver, and bronze,

with a purchase limit per box type. Minting this box requires whitelist registration and the whitelist records which box types are available for purchase and whether they have been purchased. Only one minting is allowed per whitelisted account, and there is a wallet that can mint an unlimited number of SILVER boxes. There is also a feature to open BOX NFT. Which burns BOX NFT and mints random heroes according to the probability set by box type.

NyxMultiSend

This contract has been used in the past. A function to batch send tokens to multiple users is implemented.

TestNFT

This contract is the ERC721 NFT. Anyone can mint an NFT and set the URI.

Scope

```
├── KingdomStoryBKT.sol
├── KingdomStoryHeroNFT.sol
├── KingdomStoryKID.sol
├── KingdomStoryMarket.sol
├── KingdomStoryMysteryBox.sol
├── NyxMultiSend.sol
├── TestNFT.sol
├── interfaces
│   ├── IKingdomStoryHeroNFT.sol
│   ├── IKingdomStoryMysteryBox.sol
│   ├── ILockable.sol
│   ├── IPlatformFee.sol
│   ├── ISoftnyxContract.sol
│   └── ISoftnyxMarket.sol
├── libs
│   └── TransferLib.sol
└── multi
    ├── Escapable.sol
    ├── Migrations.sol
    ├── MultiTransfer.sol
    ├── MultiTransferEtherToken.sol
    └── MultiTransferToken.sol
```

Access Controls

The access control of the audit scope refers to the following actors.

- proxy-admin
- onlyMaster(Owner + Master)
- Master
- Market
- Box

proxy-admin : The proxy-admin has permission to upgrade the proxy contract.

onlyMaster : onlyMaster verifies that msg.sender is either owner or master. The onlyMaster has the power to set key information in the contract or mint tokens.

Master : The Master has permission to set URIs in the KingdomStoryHeroNFT contract or create signatures to mint tokens.

Market : The Market has permission to transfer tokens from the KingdomStoryBKT, KingdomStoryKID, and KingdomStoryHeroNFT contracts without approval.

Box : The Box has permission to mint tokens in the KingdomStoryHeroNFT contract.

FINDINGS

1. KingdomStoryMarket design issue

ID: SOFTNYX-KINGDOM-01
Severity: High
Type: Design
Difficulty: Low

Issue

There are three design issues in the Market.

- **TradeNFT - No validation of the used check**

There is no usage validation for signature messages used in TRADE, which causes the following two problems.

a. If the signature of a sales message created by a MARKET user is exposed to the outside, the sales message cannot be canceled.
b. There is no function to record the usage of signature messages, which allows signature data to be reused.
    i. User A sells testNFT#1234 using KingdomStoryMarket.
    ii. User A buys testNFT#1234 back.
    iii. Attacker(User B) reuses the sales message and signature exposed in (i) to force the trade of User A's NFT.

- **TradeNFT - No validation of the msg.value**

Trading with a NATIVE_TOKEN does not have msg.value validation.

```
if (_currency != NATIVE_TOKEN) {
    require(msg.value == 0, "ERR_VALUE_NOT_ZERO");
    _validateERC20BalAndAllowance(
        msg.sender,
        _currency,
        _currencyAmount
    );
}
```

[https://github.com/softnyx/kingdomNFT.solidity/blob/f120792a94e0d4bf1eeda8f261b99c2838822197/contracts/KingdomStoryMarket.sol#L166-L174]

It causes two problems.

a. A larger amount of native tokens than the trade amount may be sent and accumulated the native token in the market contract.
b. If the balance is accumulated in the market contract due to (a) or the receive function in the contract, the native coin can be stolen using the TradeNFT function.

   Exploit scenario) The attacker uses two accounts, A and B.
   account A, creates a signature to sell NFTs for native coins and the balance accumulated in the market contract.
   account B, to call the TradeNFT function without msg.value.

- **master account used for multiple purposes**

The master account is being used for two purposes: Transaction Sender (onlyMaster modifier) and Message Signer (KingdomStoryHeroNFT).

As transactions are signed the same way as messages, values may be used in messages due to future feature development. Currently, when the master account is used to Message Signer, the ECDSA.toEthSignedMessageHash function is called. So, there are no issues due to the different signatures and data structures used for Transactions.

Recommendation

1. Validate signature usage based on values unique to each trade (e.g. message hash) to ensure that they are recorded and cannot be reused.
2. Implement the function to revoke a trade signature.
3. Add code to validate that _currencyAmount and msg.value are equal when native token.
4. Ensure the master account is only used for transactions to prevent potential threats. We recommended creating and managing a separate account used for UnlockAndSetTokenURI/MintHero signing in the KingdomStoryHeroNFT contract.

Fix Comment

Fixed in 6ee321f, 2973d37 and c320f79 commits. The usedSignatures validation and NATIVE_TOKEN value validation has been added. The master account and signer account are separated by using a new officialSigner account. The issue where the officialSigner initial value was the same as master has also been fixed in 349c880 commit.

2. Insufficient data in sign messages leads to potential signature issues

ID: SOFTNYX-KINGDOM-02
Severity: Low
Type: Input Validation
Difficulty: Low

Issue

The lack of information in the signature messages used in kingdomNFT contracts can lead to three potential security issues.

- The signing message does not contain information about the function for which the signature will be used. So, it could lead to the signing message being used in unintended functions.
- The signing message does not contain information about the contract for which the signature will be used. So, it could lead to the signing message being used in unintended contracts.
- The signing message does not contain information about the chain where the contract for which the signature will be used exists. The signing message can be reused if the dApp supports multichain and the contracts are deployed to the same address.

In addition, when users using kingdomNFT contracts sign a message using web3 wallet, they can't see the exact information about the message in the signature window of web3 wallet. Users can't see information about the chain where the signature will be used, the contract where the signature will be used, or the function where the signature will be used.

```
bytes32 messageHash = keccak256(
    abi.encodePacked(
        _seller,
        _currency,
        _currencyAmount,
        _nftAddr,
        _tokenid
    )
);
```

[https://github.com/softnyx/kingdomNFT.solidity/blob/f120792a94e0d4bf1eeda8f261b99c2838822197/contracts/KingdomStoryMarket.sol#L201-L209]

```
string memory str = string(
    abi.encodePacked(
        uri,
        "/",
        Strings.toString(_addressCount[_msgSender()])
    )
);
```

[https://github.com/softnyx/kingdomNFT.solidity/blob/f120792a94e0d4bf1eeda8f261b99c2838822197/contracts/KingdomStoryHeroNFT.sol#L127-L133]

```
string memory str = string(
    abi.encodePacked(
        uri,
        "/",
        Strings.toString(_addressCount[_msgSender()])
    )
);
```

[https://github.com/softnyx/kingdomNFT.solidity/blob/f120792a94e0d4bf1eeda8f261b99c2838822197/contracts/KingdomStoryHeroNFT.sol#L263-L269]

Recommendation

The signature message should include information about the function where the signature will be used (e.g., function signature), contract details (contract address), and chain information (e.g. chainId). By applying the EIP-712, which includes the chainId, contract address, and TYPEHASH (function signature), this issue can be addressed.

Fix Comment

Fixed in these two commits. EIP-712 implemented in KingdomStoryMarket and KingdomStoryHero NFT contracts.

3. KingdomStoryMysteryBox unsafe random value

ID: SOFTNYX-KINGDOM-03
Severity: Low
Type: Random Value
Difficulty: Medium

Issue

The random value used by the KingdomStoryMysteryBox contract can be predictable.

```
function _random() internal view returns (uint) {
    return
        uint(
            keccak256(
                abi.encodePacked(
                    block.timestamp,
                    block.timestamp,
                    _nonce,
                    _msgSender()
                )
            )
        );
}
```

[https://github.com/softnyx/kingdomNFT.solidity/blob/f120792a94e0d4bf1eeda8f261b99c2838822197/contracts/KingdomStoryMysteryBox.sol#L274-L286]

[attack with CA - pseudo code]

```
contract Exploit {
    uint logTokenId;

    function attack() external {
        KingdomStoryMysteryBox(...).OpenBox(tokenid);

        require(KNFTContract(...).tokenURI(logTokenId) == targetToken);
    }

     function onERC721Received(
        address operator,
        address from,
        uint256 tokenId,
        bytes calldata data
    ) external returns (bytes4) {
        logTokenId = tokenId;
    }
}
```

[Example]

[attack with EOA - exploit scenario]

```
mev_transaction_bundle([
    txOpenBox,
    txCheckTokenUri,
])
```

[Example]

Recommendation

We recommend not relying on blockchain-based information for generating random values, as they can be predictable. Instead, utilize a commit-reveal scheme or a VRF (Verifiable Random Function) to ensure true randomness.

Fix Comment

Fixed in multiple commits. We reviewed issue-3 based on the e38c5e9 commit and found an additional issue with the HeroNFT lock feature. The additional issue was resolved in the e991370 commit.

Notice: *should monitor the remaining balance in the VRF Coordinator subscription and check the hash key and contract address according to deployment environment*

4. UnlockAndBurn function without owner authentication implemented

ID: SOFTNYX-KINGDOM-04
Severity: High
Type: Access & Privilege Control
Difficulty: Low

Issue

Owner authentication does not exist in the UnlockAndBurn function, so anyone can burn another user's locked KingdomStoryHeroNFT.

```
function UnlockAndBurn(uint tokenId) public {
    require(_exists(tokenId), "ERR_NONEEXISTENT_TOKEN");
    require(unlockers[tokenId] != address(0), "ERR_NOT_LOCKED");

    if (getLocked(tokenId) != address(0)) {
        delete unlockers[tokenId];
        emit Unlock(tokenId);
    }

    _burn(tokenId);
}
```

[https://github.com/softnyx/kingdomNFT.solidity/blob/f120792a94e0d4bf1eeda8f261b99c2838822197/contracts/KingdomStoryHeroNFT.sol#L146-L156]

Recommendation

We recommend adding owner authentication to the UnlockAndBurn function.
e.g. *`require(_isApprovedOrOwner(_msgSender(), tokenId), "ERC721: caller is not token owner or approved");`*

Fix Comment

Fixed in this commit.

5. KingdomStoryHeroNFT contract reentrancy

ID: SOFTNYX-KINGDOM-05
Severity: High
Type: Reentrancy
Difficulty: Low

Issue

A reentrancy issue in the MsSafeMint, MsSafeMintMulti, SafeMintBoxResult, and MintHero functions of the KingdomStoryHero NFT contract leads to several possible attacks.

The MsSafeMint, MsSafeMintMulti, SafeMintBoxResult, and MintHero functions of the KingdomStoryHeroNFT contract use the _safeMint function internally. The _safeMint function has a callback function (onERC721Received), and an external call occurs when the callback is called. As a result, the external call can occur while executing a function of the KingdomStoryHeroNFT contract, and the function of the KingdomStoryHeroNFT contract can be executed again inside the external call.

The following attacks are possible through the reentrancy issue in the KingdomStoryHeroNFT contract.

- DoS: If the toAddress of the MsSafeMintMulti function contains a contract account (CA), it is possible to call the KingdomStoryHeroNFT.MintHero function from inside the callback function of the CA. In this case, the tokenId value and tokenIdCounter value that are being incremented in the for statement inside the MsSafeMintMulti function will become inconsistent, causing a revert when the _safeMint function is called next.
- Reuse Signature: The MintHero function works by using the signature of masterAddr, and it is implemented in such a way that the signature cannot be reused by having a "_addressCount" variable that is incremented sequentially for each account. However, since the value of _addressCount is incremented after calling the _safeMint function, if the CA calls the MintHero function and the MintHero function is called again inside the CA's callback function, the value of _addressCount is not incremented. This allows for signature reuse.

Recommendation

We recommend using the ReentrancyGuardUpgradeable.nonReentrant modifier in functions where _safeMint is utilized to prevent reentrancy issues. Additionally, we recommend restructuring the MintHero function according to the checks-effects-interactions pattern. (_addressCount is incremented immediately after signature verification.)

Fix Comment

Fixed in this commit.

6. KingdomStoryHeroNFT signature reuse issue 1

ID: SOFTNYX-KINGDOM-06
Severity: Medium
Type: Input Validation
Difficulty: Low

Issue

An attacker can reuse a signature issued to another user in the UnlockAndSetTokenURI / MintHero function. The UnlockAndSetTokenURI/MintHero function works by using the signature of the master. The signature data includes a URI, and the addressCount value that exists for each Account. Since there is no information (address) in the signature data that can identify an Account, the attacker can reuse signatures issued to other users.

```
function UnlockAndSetTokenURI(
    uint256 tokenId,
    string memory uri,
    bytes memory signature
) public virtual onlyUnlocker(tokenId) {
    require(unlockers[tokenId] != address(0), "ERR_NOT_LOCKED");
    string memory str = string(
        abi.encodePacked(
            uri,
            "/",
            Strings.toString(_addressCount[_msgSender()])
        )
    );

    require(
        _recoverSigner(bytes(str), signature) == masterAddr,
        "ERR_HASH_NOT_MATCH"
    );
```

[https://github.com/softnyx/kingdomNFT.solidity/blob/f120792a94e0d4bf1eeda8f261b99c2838822197/contracts/KingdomStoryHeroNFT.sol#L121-L138]

```
function MintHero(string memory uri, bytes memory signature) public {
    string memory str = string(
        abi.encodePacked(
            uri,
            "/",
            Strings.toString(_addressCount[_msgSender()])
        )
    );
    require(
        _recoverSigner(bytes(str), signature) == masterAddr,
        "ERR_HASH_NOT_MATCH"
    );
```

[https://github.com/softnyx/kingdomNFT.solidity/blob/f120792a94e0d4bf1eeda8f261b99c2838822197/contracts/KingdomStoryHeroNFT.sol#L262-L273]

Recommendation

We recommend including the Account Address information in the signature data.

Fix Comment

Fixed in this commit. The address information has been added.

7. KingdomStoryHeroNFT signature reuse issue 2

ID: SOFTNYX-KINGDOM-07
Severity: Medium
Type: Input Validation
Difficulty: Low

Issue

An attacker can reuse the signature in different functions of UnlockAndSetTokenURI / MintHero. The UnlockAndSetTokenURI/MintHero function works by using the master's signature. The signature can be reused in both functions because no information exists about what the signature data is used for, and the signature data structure is the same for both functions.

```
function UnlockAndSetTokenURI(
    uint256 tokenId,
    string memory uri,
    bytes memory signature
) public virtual onlyUnlocker(tokenId) {
    require(unlockers[tokenId] != address(0), "ERR_NOT_LOCKED");
    string memory str = string(
        abi.encodePacked(
            uri,
            "/",
            Strings.toString(_addressCount[_msgSender()])
        )
    );

    require(
        _recoverSigner(bytes(str), signature) == masterAddr,
        "ERR_HASH_NOT_MATCH"
    );
```

[https://github.com/softnyx/kingdomNFT.solidity/blob/f120792a94e0d4bf1eeda8f261b99c2838822197/contracts/KingdomStoryHeroNFT.sol#L121-L138]

```
function MintHero(string memory uri, bytes memory signature) public {
    string memory str = string(
        abi.encodePacked(
            uri,
            "/",
            Strings.toString(_addressCount[_msgSender()])
        )
    );
    require(
        _recoverSigner(bytes(str), signature) == masterAddr,
        "ERR_HASH_NOT_MATCH"
    );
```

[https://github.com/softnyx/kingdomNFT.solidity/blob/f120792a94e0d4bf1eeda8f261b99c2838822197/contracts/KingdomStoryHeroNFT.sol#L262-L273]

Recommendation

We recommend adding the purpose of the signature into the signature data. For example, function signature 4-byte can be prefixed.

Fix Comment

Fixed in this commit. The TYPEHASH information has been added.

8. KingdomStoryHeroNFT-UnlockAndSetTokenURI insufficient tokenId validation

ID: SOFTNYX-KINGDOM-08
Severity: Low
Type: Input Validation
Difficulty: Low

Issue

An attacker can overwrite arbitrary token URI information in the KingdomStoryHeroNFT contract.

The signature data for the UnlockAndSetTokenURI function of the KingdomStoryHeroNFT contract is the uri and the addressCount value that exists for each account. Since the signature data does not include the information of the tokenId to be modified, it is possible for the master to change the URI information of a token other than the intended token.

```
function UnlockAndSetTokenURI(
    uint256 tokenId,
    string memory uri,
    bytes memory signature
) public virtual onlyUnlocker(tokenId) {
    require(unlockers[tokenId] != address(0), "ERR_NOT_LOCKED");
    string memory str = string(
        abi.encodePacked(
            uri,
            "/",
            Strings.toString(_addressCount[_msgSender()])
        )
    );
```

[https://github.com/softnyx/kingdomNFT.solidity/blob/f120792a94e0d4bf1eeda8f261b99c2838822197/contracts/KingdomStoryHeroNFT.sol#L121-L138]

Recommendation

We recommend including the tokenId value in the signature data.

Fix Comment

Fixed in this commit. The tokenId information has been added.

9. Upgradeable issue

ID: SOFTNYX-KINGDOM-09
Type: Off-standard
Severity: Tips
Difficulty: N/A

Issue

The KingdomStoryBKT, KingdomStoryHeroNFT, KingdomStoryKID, KingdomStoryMarket, and KingdomStoryMysteryBox contracts use the OpenZeppelin Initializable module. However, they are not calling the _disableInitializers function in the constructor of the Implementation contract.

If the _disableInitializers function is not called when deploying the Implementation contract, an unauthorized user can call the initialize function of the implementation contract.

Recommendation

We recommend adding a constructor[1] function to the KingdomStoryBKT, KingdomStoryHeroNFT, KingdomStoryKID, KingdomStoryMarket, and KingdomStoryMysteryBox contracts.

Fix Comment

Fixed in this commit.

[1] https://docs.openzeppelin.com/upgrades-plugins/1.x/writing-upgradeable#initializing_the_implementation_contract

10. function/variables naming style

ID: SOFTNYX-KINGDOM-10
Severity: Tips
Type: Off-standard
Difficulty: N/A

Issue

The kingdomNFT contracts have function names and variable names that violate solidity's naming style.

[Function Name]	[Variable Name]
KingdomStoryBKT.MsSetMarketAddr KingdomStoryBKT.MsSetMasterAddr KingdomStoryHeroNFT.MsSetMarketAddr KingdomStoryHeroNFT.MsSetMasterAddr KingdomStoryHeroNFT.MsSetBoxAddr KingdomStoryHeroNFT.GetBoxAddr KingdomStoryHeroNFT.UnlockAndSetTokenURI KingdomStoryHeroNFT.UnlockAndBurn KingdomStoryHeroNFT.MsSetTokenURI KingdomStoryHeroNFT.MsPause KingdomStoryHeroNFT.MsUnpause KingdomStoryHeroNFT.MsBurnMany KingdomStoryHeroNFT.MsSafeMint KingdomStoryHeroNFT.MsSafeMintMulti KingdomStoryHeroNFT.SafeMintBoxResult KingdomStoryHeroNFT.MintHero KingdomStoryHeroNFT._recoverSigner KingdomStoryHeroNFT.MsGetAddressCount KingdomStoryKID.MsSetMarketAddr KingdomStoryKID.MsSetMasterAddr KingdomStoryMarket.MsSetMasterAddr KingdomStoryMarket.MsSetNftAddress KingdomStoryMarket.MsSetBoxAddress KingdomStoryMarket.TradeNFT KingdomStoryMarket.Verify KingdomStoryMarket._recoverSigner KingdomStoryMarket.GetContractBalance KingdomStoryMysteryBox.MsSetMasterAddr KingdomStoryMysteryBox.MsSetNftContract KingdomStoryMysteryBox.MsSetBoxIpfsHash KingdomStoryMysteryBox.MsSetItemHash KingdomStoryMysteryBox.MsSetLimitCount KingdomStoryMysteryBox.MsSetWhiteList KingdomStoryMysteryBox.CheckAddrStatus KingdomStoryMysteryBox.BuyBox KingdomStoryMysteryBox.OpenBox KingdomStoryMysteryBox.GetBuyCount KingdomStoryMysteryBox.GetLimitCount KingdomStoryMysteryBox.MsGetHashCount KingdomStoryMysteryBox.MsGetWhiteListCount KingdomStoryMysteryBox.MsSetMarketAddress KingdomStoryMysteryBox.MsSetUnCountAddress TestNFT.SetTokenURI IKingdomStoryHeroNFT.SafeMintBoxResult ISoftnyxContract.MsSetMasterAddr	KingdomStoryHeroNFT._addressCount KingdomStoryMarket.NFTADDR KingdomStoryMarket.BOXADDR KingdomStoryMysteryBox.UnCountBuyer KingdomStoryMysteryBox.NftContracts KingdomStoryMysteryBox.BoxTypes KingdomStoryMysteryBox._whiteList KingdomStoryMysteryBox._boxHash KingdomStoryMysteryBox._itemHash KingdomStoryMysteryBox._limitCount KingdomStoryMysteryBox._buyCount KingdomStoryMysteryBox._whiteListCount

Recommendation

We recommend adhering to Solidity's naming conventions[2] and updating the variable and function names accordingly.

Fix Comment

The issue is acknowledged.

[2] https://docs.soliditylang.org/en/v0.8.21/style-guide.html#naming-styles

11. unused functions

ID: SOFTNYX-KINGDOM-11
Severity: Tips
Type: Incorrect Interface
Difficulty: N/A

Issue

An unused mint function exists for the KingdomStoryHeroNFT and KingdomStoryMysteryBox contracts.

```
function mint(uint tokenid) public view onlyMaster {
    require(tokenid == 0, "");
    require(false, "ERR_BANNED");
}
```

[https://github.com/softnyx/kingdomNFT.solidity/blob/f120792a94e0d4bf1eeda8f261b99c2838822197/contracts/KingdomStoryHeroNFT.sol#L179-L182]

```
function mint(uint tokenid) public view onlyMaster {
    require(tokenid == 0, "");
    require(false, "ERR_BANNED");
}
```

[https://github.com/softnyx/kingdomNFT.solidity/blob/f120792a94e0d4bf1eeda8f261b99c2838822197/contracts/KingdomStoryMysteryBox.sol#L151-L154]

Recommendation

We recommend removing the unused mint functions.

Fix Comment

Fixed in this commit.

12. non-implemented pausable functions

ID: SOFTNYX-KINGDOM-12
Type: Incorrect Interface
Severity: Tips
Difficulty: N/A

Issue

The KingdomStoryMysteryBox contract inherited the PausableUpgradeable contract and uses the whenNotPaused modifier in the _beforeTokenTransfer function, but the pause/unpause function is not implemented.

Recommendation

We recommend implementing the pause and unpause functions in the KingdomStoryMysteryBox contract.

Fix Comment

Fixed in this commit.

13. Missing event logging in several functions

ID: SOFTNYX-KINGDOM-13
Type: Visibility
Severity: Tips
Difficulty: N/A

Issue

Important functions that set and modify master and contracts{nft/box/market} information do not emit events. Without event-based logging, transaction auditing and incident response can become more difficult. A list of functions that are missing events:

- *.initialize
- *.MsSetMasterAddr
- KingdomStoryBKT.MsSetMarketAddr
- KingdomStoryMarket.MsSetNftAddress
- KingdomStoryMarket.MsSetBoxAddress
- KingdomStoryMarket.MsSetNftAddress
- KingdomStoryKID.MsSetMarketAddr
- KingdomStoryMysteryBox.MsSetNftContract
- KingdomStoryMysteryBox.MsSetMarketAddress

Recommendation

We recommend adding features to emit the event in the problematic functions.

Fix Comment

Fixed in 985aad1, fca7c63 commits.

14. Incorrect implementation due to mixing of token value and native-coin value in multiTransferEther function of MultiTransferEtherToken

ID: SOFTNYX-KINGDOM-14
Severity: Critical
Type: Input Validation
Difficulty: Low

Issue

The multiTransferEther function incorrectly validates the amount of native coins transferred, which could lead to the theft of the native coins in the NyxMultiSend (MultiTransferEtherToken) contract.

The multiTransferEther function of the MultiTransferEtherToken contract takes two values, token amounts (_amounts) and native-coin amounts (_amountsEther), and transfers tokens and native coins. The native coin incorrectly validates the amount transferred, which allows the native coin of the NyxMultiSend (MultiTransferEtherToken) contract to be stolen.

```
function multiTransferEther(address _token, address payable[] calldata _addresses, uint256[] calldata
_amounts, uint256 _amountSum, uint256[] calldata _amountsEther) external whenNotPaused payable {
    uint256 _value = msg.value;
    IERC20 token = IERC20(_token);
    token.safeTransferFrom(msg.sender, address(this), _amountSum);

    for (uint8 i; i < _addresses.length; i++)
    {
        _amountSum = _amountSum.sub(_amounts[i]);
        _value = _value.sub(_amounts[i]);
        token.transfer(_addresses[i], _amounts[i]);

        _addresses[i].transfer(_amountsEther[i]);
    }
}
```

[https://github.com/softnyx/kingdomNFT.solidity/blob/f120792a94e0d4bf1eeda8f261b99c2838822197/contracts/multi/MultiTransferEtherToken.sol#L13-L26]

Recommendation

The _amountsEther[i] value must be subtracted from the _value value.

```
diff --git a/contracts/multi/MultiTransferEtherToken.sol b/contracts/multi/MultiTransferEtherToken.sol
index 68cdf19..af4c5f2 100644
--- a/contracts/multi/MultiTransferEtherToken.sol
+++ b/contracts/multi/MultiTransferEtherToken.sol
@@ -18,7 +18,7 @@ contract MultiTransferEtherToken is Pausable {
         for (uint8 i; i < _addresses.length; i++)
         {
             _amountSum = _amountSum.sub(_amounts[i]);
-            _value = _value.sub(_amounts[i]);
+            _value = _value.sub(_amountsEther[i]);
             token.transfer(_addresses[i], _amounts[i]);

             _addresses[i].transfer(_amountsEther[i]);
```

[Example]

Fix Comment

Fixed in this commit.

15. KingdomStoryMysteryBox contract reentrancy

ID: SOFTNYX-KINGDOM-11
Severity: High
Type: Reentrancy
Difficulty: Low

Issue

In the KingdomStoryMysteryBox contract, boxes can be used for a higher tier than the one purchased.

The BuyBox function of the KingdomStoryMysteryBox contract works in the following order: _safeMint -> _setTokenURI -> set BoxType.

```
function BuyBox(BoxType boxType) external payable {
     ...
     if (
         boxType == BoxType.Gold ||
         boxType == BoxType.Silver ||
         boxType == BoxType.Bronze
     ) {
         if (boxType != BoxType.Silver || _msgSender() != UnCountBuyer) {
             _whiteList[boxType][_msgSender()] = 2; // 이미 민팅한 유저는 2로 고정
             _buyCount[boxType]++;
         }

         uint256 tokenId = _tokenIdCounter.current();
         _tokenIdCounter.increment();

         _safeMint(_msgSender(), tokenId);
         _setTokenURI(tokenId, _boxHash[boxType]);

         BoxTypes[tokenId] = boxType;
     }
 }
```

[https://github.com/softnyx/kingdomNFT.solidity/blob/f120792a94e0d4bf1eeda8f261b99c2838822197/contracts/KingdomStoryMysteryBox.sol#L156-L185]

The _safeMint function has a callback function (onERC721Received) and an external call occurs when the callback is called. As a result, the external call occurs while executing a function of the KingdomStoryMysteryBox contract, and the function of the KingdomStoryHeroNFT contract can be executed again inside the external call.

There are 6 BoxTypes in the KingdomStoryMysteryBox contract, with the Gold Type starting at 0.

```
enum BoxType {
     Gold,
     Silver,
```

```
    Bronze,
    MembershipGold,
    MembershipSilver,
    MembershipBronze
}
```

[https://github.com/softnyx/kingdomNFT.solidity/blob/f120792a94e0d4bf1eeda8f261b99c2838822197/contracts/interfaces/IKingdomStoryMysteryBox.sol#L7-L14]

Suppose the _safeMint function is executed in the BuyBox function of the KingdomStoryMysteryBox contract, and an attacker calls the OpenBox function of the KingdomStoryMysteryBox contract inside the external call. In that case, the attacker can use the NFT without setting BoxType to be executed. If the set BoxType is not executed, the attacker will get 0 (Gold) when looking up the BoxType, and in the OpenBox function the attacker can use the Box with Gold type even if purchased Silver / Bronze in the BuyBox.

Recommendation

The following three recommendations should be implemented:

1. Add the nonReentrant modifier to the BuyBox/OpenBox function.
2. Change the OpenBox function flow to the checks-effects-interactions pattern. Change the order to call _safeMint after setting the BoxType.
3. Add a None Type to the zero of the BoxType to prevent it from being recognized as a BoxType when it is not initialized.

Fix Comment

Fixed in this commit. All three recommendations have been applied.

DISCLAIMER

This report does not guarantee investment advice, the suitability of the business models, and codes that are secure without bugs. This report shall only be used to discuss known technical issues. Other than the issues described in this report, undiscovered issues may exist such as defects on the main network. In order to write secure codes, correction of discovered problems and sufficient testing thereof are required.

Appendix. A

Severity Level

CRITICAL	Must be addressed as a vulnerability that has the potential to seize or freeze substantial sums of money.
HIGH	Has to be fixed since it has the potential to deny users compensation or momentarily freeze assets.
MEDIUM	Vulnerabilities that could halt services, such as DoS and Out-of-Gas, need to be addressed.
LOW	Issues that do not comply with standards or return incorrect values
TIPS	Tips that makes the code more usable or efficient when modified

Difficulty Level

	Low	Medium	High
Privilege	anyone	Miner/Block Proposer	Admin/Owner
Capital needed	Small or none	Gas fee or volatile as price change	More than exploited amount
Probability	100%	Depend on environment	Hard as mining difficulty

Vulnerability Category

Arithmetic	• Integer under/overflow vulnerability • floating point and rounding accuracy
Access & Privilege Control	• Manager functions for emergency handle • Crucial function and data access • Count of calling important task, contract state change, intentional task delay
Denial of Service	• Unexpected revert handling • Gas limit excess due to unpredictable implementation
Miner Manipulation	• Dependency on the block number or timestamp. • Frontrunning
Reentrancy	•Proper use of Check-Effect-Interact pattern. •Prevention of state change after external call • Error handling and logging.
Low-level Call	• Code injection using delegatecall • Inappropriate use of assembly code
Off-standard	• Deviate from standards that can be an obstacle of interoperability.
Input Validation	• Lack of validation on inputs.
Logic Error/Bug	• Unintended execution leads to error.
Documentation	•Coherency between the documented spec and implementation
Visibility	• Variable and function visibility setting
Incorrect Interface	• Contract interface is properly implemented on code.

End of Document