Vabble Audit Report

PeckShield

October 3, 2023

PVE001: Reentrancy Risk Avoidance in FactoryFilmNFT/FactorySubNF

```
/// @notice User mint the subscription NFTs to "_to" address
function mint(
    address _token,
    address _to,
    uint256 _subPeri
    uint256 _category
) public payable {
    if(_token != IOwnablee(OWNABLE).PAYOUT_TOKEN() && _token != add
        require(IOwnablee(OWNABLE).isDepositAsset(_token), "mint: n
    }
    require(subNFTAddress != address(0), "mint: not deploy yet");
    require(mintInfo[_category].maxMintAmount > 0, "mint: no admin
    require(mintInfo[_category].maxMintAmount > getTotalSupply(), "

    __handleMintPay(_token, _subPeriod, _category);
```

Re-enter mint()

```
function __handleMintPay(...
) private {
    uint256 price = mintInfo[_category].mintPrice;
    uint256 expectAmount = getExpectedTokenAmount(_payToken, _period *
    // Return remain ETH to user back if case of ETH and Transfer Asset
    if(_payToken == address(0)) {
        require(msg.value >= expectAmount, "handlePay: Insufficient pai
        if (msg.value > expectAmount) {
            Helper.safeTransferETH(msg.sender, msg.value - expectAmount
        }
```

- Bypass maxMintAmount Limit

PVE002: Possible Sandwich/MEV For Reduced Returns

```
/// @notice Swap eth/token to another token
function swapAsset(bytes calldata _swapArgs) external transferHandler(_swapArgs) returns (uint256 amount
    ( ... ) = abi.decode(_swapArgs, (uint256, address, address));

    (address router, , address weth, address[] memory path) = __checkPool(depositAsset, incomingAsset);
    require(router != address(0), "swapAsset: No Pool");

    // Get payoutAmount from depositAsset on Uniswap
    uint256 expectAmount = IUniswapV2Router(router).getAmountsOut(depositAmount, path)[1];

    if(path[0] == weth) {
        amount_ = __swapETHToToken(depositAmount, expectAmount, router, path)[1];
    } else {
        amount_ = __swapTokenToToken(depositAmount, expectAmount, router, path)[1];
    }
}
```

The expectAmount calculation is affected by MEV

- Example Sandwich
  - First, huge swap (0 -> 1) to inflate reserveA and reduce expectAmount
  - Next, swapAsset(): swaps 0 -> 1; Then reserve swap (1 -> 0) to profit

PVE003: No ETH Support in FactoryFilmNFT::mintToBatch()

```
function mintToBatch(
    uint256[] calldata _filmIdList,
    address[] calldata _toList,
    address _payToken
) external {
    require(_toList.length > 0, "mintBatch: zero item length")
    require(_toList.length == _filmIdList.length, "mintBatch:
    require( _payToken != address(0) );
    for(uint256 i; i < _toList.length; i++) {
        mint(_filmIdList[i], _toList[i], _payToken);
    }
}

function mint(
    uint256 _filmId,
    address _to,
    address _payToken
) public payable {
    if(_payToken != IOwnablee(OWNABLE).PAYOUT_TOKEN() && _payT
```

- Otherwise, not compatible with __handleMintPay()
  - Same for FactorySubNFT::mintToBatch()

PVE004: Collusion-Based Revenue Collection w/ JIT Film NFTs

```
// Transfer revenue amount to user if user fund to this film throughout NFT mint
function __payRevenue(
    address _user,
    address _vabToken,
    uint256 _filmId,
    uint256 _payout
) private {
    uint256 nftCountOwned;
    uint256[] memory nftList = IFactoryFilmNFT(FILM_NFT_FACTORY).getFilmTokenIdList(_filmId);
    for(uint256 i = 0; i < nftList.length; i++) {
        if(IERC721(FILM_NFT_FACTORY).ownerOf(nftList[i]) == _user) nftCountOwned += 1;
    }

    ( , , , , uint256 revenuePercent, ,) = IFactoryFilmNFT(FILM_NFT_FACTORY).getMintInfo(_filmId);
    uint256 revenueAmount = nftCountOwned * _payout * revenuePercent / 1e10;
    if(_payout >= revenueAmount && revenueAmount > 0) {
        require(StudioPool >= revenueAmount, "revenue: insufficient studio pool");
        require(IERC20(_vabToken).balanceOf(address(this)) >= revenueAmount, "revenue: insufficient balance");

        Helper.safeTransfer(IOwnablee(OWNABLE).PAYOUT_TOKEN(), _user, revenueAmount);
        StudioPool -= revenueAmount;
    }
}
```

Collect Just-In-Time Film NFTs right before revenue payment

PVE005: Enforcement of One-Time Initialization in FactoryFilmNFT

FactoryFilmNFT.sol

```
101     /// @notice onlyStudio set mint info for his films
102     // maxMintCount * (mintPrice - mintPrice * feePercent) > fundRaiseAmount
103     function setMintInfo(
110     ) external {
111         require(_amount > 0 && _price > 0 && _tier > 0, "setMint: Zero value");
112         require(_feePercent <= IProperty(DAO_PROPERTY).maxMintFeePercent(), "setM
113         require(_revenuePercent < 1e10, "setMint: over 100%");
114
115         address owner = IVabbleDAO(VABBLE_DAO).getFilmOwner(_filmId);
116         require(owner == msg.sender, "setMint: not film owner");
117
118         (uint256 raiseAmount, , uint256 fundType) = IVabbleDAO(VABBLE_DAO).getFil
119         if(fundType > 0) { // case of funding film
120             require(_amount * _price * (1e10 - _feePercent) / 1e10 > raiseAmount,
121         }
122
123         Mint storage mInfo = mintInfo[_filmId];
124         mInfo.tier = _tier;                                    // 1, 2, 3
```

Should be initialized only once

require(mintInfo[_filmId].amount=0);

PVE005-2: Enforcement of One-Time Initialization in FactoryFilmNFT

```
/// @notice Studio deploy a nft contract per filmId
function deployFilmNFTContract( ...
) external nonReentrant {
    require(IVabbleDAO(VABBLE_DAO).getFilmOwner(_filmId) == msg.sender, "deployNFT: not film c

    (, uint256 fundPeriod, uint256 fundType) = IVabbleDAO(VABBLE_DAO).getFilmFund(_filmId);
    require(fundType == 2 || fundType == 3, "deployNFT: not fund type by NFT");

    Helper.Status status = IVabbleDAO(VABBLE_DAO).getFilmStatus(_filmId);
    require(status == Helper.Status.APPROVED_FUNDING, "deployNFT: filmId not approved for func

    (, uint256 pApproveTime) = IVabbleDAO(VABBLE_DAO).getFilmProposalTime(_filmId);
    require(fundPeriod >= block.timestamp - pApproveTime, "deployNFT: passed funding period");

    VabbleNFT t = new VabbleNFT(baseUri, collectionUri, _name, _symbol, address(this));
    filmNFTContract[_filmId] = t;
    require( mintInfo[_filmId].nft=address(0) );
    Mint storage mInfo = mintInfo[_filmId];
    mInfo.nft = address(t);
```

Should be initialized only once

PVE006: Improper Update on Film Fund Raise in FactoryFilmNFT

```
function mint(
    uint256 _filmId,
    address _to,
    address _payToken
) public payable {
    if(_payToken != IOwnablee(OWNABLE).PAYOUT_TOKEN() && _payToken != address(0)) {
        require(IOwnablee(OWNABLE).isDepositAsset(_payToken), "mint: not allowed asset");
    }
    require(mintInfo[_filmId].maxMintAmount > 0, "mint: no mint info");
    require(mintInfo[_filmId].maxMintAmount > getTotalSupply(_filmId), "mint: exceed mint amount");

    __handleMintPay(_filmId, _payToken);
    fileFundRaisedByNFT[_fileId] += mintInfo[_filmId].price;
    VabbleNFT t = filmNFTContract[_filmId];
    uint256 tokenId = t.mintTo(_to);
    filmNFTTokenList[_filmId].push(tokenId);

    emit FilmERC721Minted(address(t), tokenId, _to, block.timestamp);
}
```

PVE007: Trust Issue of Admin Keys (Questionable if EOA)

- Owner is privileged to guard/coordinate token-wide operations
  - Configure parameters, execute privileged operations, …

```
function setup(⋯
) external onlyAuditor {⋯
}

function transferAuditor(address _newAuditor) external onlyAuditor {⋯
}

function replaceAuditor(address _newAuditor) external onlyVote {⋯
}

function addDepositAsset(address[] calldata _assetList) external onlyAuditor {⋯
}

function removeDepositAsset(address[] calldata _assetList) external onlyAuditor {⋯
}
```

PVE008: Possibly Out-of-Sync Reward Boost in StakingPool

```
/// @notice Calculate reward amount without extra reward amount for listing film vote
function calcRewardAmount(address _customer) public view returns (uint256 amount_) {
    ...
    // Get proposal count started in withdrawable period of customer
    uint256 proposalCount = 0;
    for(uint256 i = 0; i < proposalCreatedTimeList.length; i++) {
        if(proposalCreatedTimeList[i] > si.stakeTime && proposalCreatedTimeList[i] < si.withdrawa
            proposalCount += 1;
        }
    }...

    // if no proposal then full rewards, if no vote for 5 proposals then no rewards, if 3 votes f
    if(proposalCount > 0) {
        if(si.voteCount == 0) {
            rewardAmount = 0;
        } else {
            uint256 countVal = (si.voteCount * 1e4) / proposalCount;
            rewardAmount = rewardAmount * countVal / 1e4;
        }
    }

    // If customer is film board member, more rewards(25%)
```

proposalCount: # proposals in stake period

voteCount: # voted proposals in lifetime

N1: Improved Logic in Ownablee::removeDepositAsset()

```
function removeDepositAsset(address[] calldata _assetList) external onlyAuditor {
    require(_assetList.length > 0, "removeDepositAsset: zero list");

    for(uint256 i = 0; i < _assetList.length; i++) {
        if(!allowAssetToDeposit[_assetList[i]]) continue;

        for(uint256 k = 0; k < depositAssetList.length; k++) {
            if(_assetList[i] == depositAssetList[k]) {
                depositAssetList[k] = depositAssetList[depositAssetList.length - 1];
                depositAssetList.pop();

                allowAssetToDeposit[_assetList[i]] = false;  break;
            }
        }

    }
}
```

N2: Removal of Unused State/Code/Event

VabbleNFT

```
function mintTo(address _to) public payable returns (uint256) {
    require(msg.sender != address(0), "mintTo: caller is zero address");
    require(msg.sender == FACTORY, "mintTo: caller is not factory contract");

    uint256 newTokenId = __getNextTokenId();
    _safeMint(_to, newTokenId);

    return newTokenId;
}
```

Redundant

StakingPool

```
/// @notice Staking VAB token by staker
function stakeVAB(uint256 _amount) external nonReentrant {
    require(isInitialized, "stakeVAB: Should be initialized");
    require(msg.sender != address(0) && _amount > 0, "stakeVAB: Zero value");

    uint256 minAmount = 10**IERC20Metadata(IOwnablee(OWNABLE).PAYOUT_TOKEN()).de
    require(_amount > minAmount, "stakeVAB: less amount than 0.01");

    Helper.safeTransferFrom(IOwnablee(OWNABLE).PAYOUT_TOKEN(), msg.sender, addre

    Stake storage si = stakeInfo[msg.sender];
    if(si.stakeAmount == 0 && si.stakeTime == 0) {
        stakerCount.increment();
    }
```

N2-2: Removal of Unused State/Code/Event

```
string public baseUri;                          // Base URI
string public collectionUri;                    // Collection URI

mapping(uint256 => Mint) private mintInfo;                  // (category => AdminMint)
mapping(uint256 => Lock) private lockInfo;            // (tokenId => SubLock)
mapping(address => uint256[]) public subNFTTokenList;    // (user => minted tokenId list)
mapping(address => address[]) public userNFTContractList; //

uint256[] public categoryList;
VabbleNFT private subNFTContract;
```

FactorySubNFT

Not used

N2-3: Removal of Unused State/Code/Event

StakingPool.sol

StakingPool

```
41      }
42
43      address private immutable OWNABLE;      // Ownablee contract address
44      address private VOTE;                   // Vote contract address
45      address private VABBLE_DAO;             // VabbleDAO contract address
46      address private VABBLE_FUNDING;         // VabbleFunding contract address
47      address private DAO_PROPERTY;           // Property contract address
48
49      uint256 public totalStakingAmount;   //
50      uint256 public totalRewardAmount;    //
51      uint256 public totalRewardIssuedAmount;    //
52      uint256 public lastfundProposalCreateTime;// funding proposal created time(
53      bool public isInitialized;           // check if contract initialized or no
54      uint256[] private proposalCreatedTimeList; // need for calculating rewards
55
```

Not used

N3: Improved Validation of Function Arguments

```
/// @notice Set subscription nft mint information by Auditor.
function setMintInfo(
    uint256 _mintAmount,
    uint256 _mintPrice,
    uint256 _lockPeriod,
    uint256 _category
) external onlyAuditor {
    require(_mintAmount > 0, "setAdminMint: zero mint amount");
    require(_category > 0, "setAdminMint: zero category");

    Mint storage amInfo = mintInfo[_category];
    amInfo.maxMintAmount = _mintAmount;
    amInfo.mintPrice = _mintPrice;
    amInfo.lockPeriod = _lockPeriod;

    categoryList.push(_category);
}
```

Validate mintInfo[_category] is empty

FactorySubNFT

N3-2: Improved Validation of Function Arguments

FactoryTierNFT

```
/// @notice onlyStudio set tier info for his films
function setTierInfo(
    uint256 _filmId,
    uint256[] calldata _minAmounts,
    uint256[] calldata _maxAmounts
) external {
    require(_minAmounts.length > 0, "setTier: bad minAmount length");
    require(_minAmounts.length == _maxAmounts.length, "setTier: bad maxAmount length");
    require(IVabbleDAO(VABBLE_DAO).getFilmOwner(_filmId) == msg.sender, "setTier: not film owner");

    (uint256 raiseAmount, uint256 fundPeriod, uint256 fundType) = IVabbleDAO(VABBLE_DAO).getFilmFund(_
    (, uint256 pApproveTime) = IVabbleDAO(VABBLE_DAO).getFilmProposalTime(_filmId);
    require(fundPeriod < block.timestamp - pApproveTime, "setTier: fund period yet");
    require(fundType > 0, "setTier: not fund film");

    uint256 raisedAmount = IVabbleFunding(FUNDING).getRaisedAmountByToken(_filmId);
    require(raisedAmount > 0 && raisedAmount >= raiseAmount, "setTier: not raised yet");

    for(uint256 i = 0; i < _minAmounts.length; i++) {
        require(_minAmounts[i] > 0, "setTier: zero value"
                                    &&( (_minAmounts[I] < maxAmounts[i])
                                    || maxAmounts[i]==0) )

        tierInfo[_filmId][i+1].minAmount = _minAmounts[i];
        tierInfo[_filmId][i+1].maxAmount = _maxAmounts[i];
    }

    tierCount[_filmId] = _minAmounts.length;
```

N3-3: Improved Validation of Function Arguments

StakingPool

```
/// @notice Initialize Vote
function initializePool(
    address _vabbleDAO,
    address _funding,
    address _property,
    address _vote
) external onlyAuditor {
    require(_vabbleDAO != address(0), "initializePool: Zero vabbleDAO address");
    VABBLE_DAO = _vabbleDAO;
    require(_funding != address(0), "initializePool: Zero funding address");
    VABBLE_FUNDING = _funding;
    require(_property != address(0), "initializePool: Zero propertyContract address")
    DAO_PROPERTY = _property;
    require(_vote != address(0), "initializePool: Zero voteContract address");
    VOTE = _vote;

    isInitialized = true;
}
```

require(!isInitialized);

- Same for Vote::initializeVote()

N4: Improved Removal of File Board Candidate

```
/// @notice Add a member to whitelist by Vote contract
function addFilmBoardMember(address _member) external onlyVote nonReentrant {
    require(isBoardWhitelist[_member] == 1, "addFilmBoardMember: Already film board

    filmBoardMembers.push(_member);
    isBoardWhitelist[_member] = 2;

    emit FilmBoardMemberAdded(msg.sender, _member, block.timestamp);
}

/// @notice Remove a member from candidate by Vote contract
function removeFilmBoardCandidate(address _member) external onlyVote nonReentrant {
    require(isBoardWhitelist[_member] == 1, "addFilmBoardMember: Already film board

    isBoardWhitelist[_member] = 0;
}
```

Property

remove member from filmBoardCandidates

- Shall we do the same in other candidates ?
  - e.g., removeRewardAddressCandidate(), removeFilmBoardCandidate()

N5: Revisited ERC2981 Use in VabbleNFT

VabbleNFT

```
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol";
import "@openzeppelin/contracts/token/common/ERC2981.sol";
import "@openzeppelin/contracts/utils/Counters.sol";
import "@openzeppelin/contracts/utils/Strings.sol";

contract VabbleNFT is ERC2981, ERC721Enumerable {

    using Counters for Counters.Counter;
    using Strings for uint256;

    Counters.Counter private nftCount;
    string public baseUri;                    // Base URI
    string public collectionUri;              // Collection URI

    address public immutable FACTORY;
```

Is royalty ever used?

N6: Revisited Function Scope of __swapETHToToken()

UniHelper

```
/// @notice Swap ETH to ERC20 Token
function __swapETHToToken(
    uint256 _depositAmount,
    uint256 _expectedAmount,
    address _router,
    address[] memory _path
) public payable returns (uint256[] memory amounts_) {
    require(address(this).balance >= _depositAmount, "swapETHToToken: Insufficient paid");

    amounts_ = IUniswapV2Router(_router).swapExactETHForTokens{value: address(this).balance}(
        _expectedAmount,
        _path,
        address(this),
        block.timestamp + 1
    );
}
```

private