Radio Caca

Contract Audit Report

VER 1.0

13th June 2022

No. 2022061316151

Project Summary

1. Project Introduction

The project based on erc721L extends the security locking mechanism of the erc721 standard. The NFT owner allows the third party to lock the NFT through <setlockapprovalforall() > or <lockapprove() >, while the authorized third party locks the NFT through <lockfrom() >. The locked NFT cannot be transferred before the end of the locking period. This enables NFT owners to participate in NFT pledge projects without transferring NFT ownership.

The project based on openpfp-contracts is an NFT market incubated by radio CACA (Raca), which issues $pfpcoin as the governance token of “openpfp.com", with a total supply of 1billion. Moreover, "openpfp.com" will provide the head trading team with the United States of Mars (USM) land. "Openpfp.com" supports Eth and stable currency as the payment currency, and 2% of the transaction fee will be used to fund the operation and reward ($pfpcoin/usdt) pledge LP.

2. Audit Summary

Project Name	Radio Caca	**Project Name**	N/A
Token	N/A	**Token**	N/A
Start date	31st May 2022	**Start date**	Solidity
End date	8th June 2022	**End date**	N/A
Github	https://github.com/radiocaca/ERC721L/blob/417f4c4bb4bb20bd14aa04865384fb90da250975/contracts/NFTs/MatrixPlusBox.sol https://github.com/radiocaca/ERC721L/blob/417f4c4bb4bb20bd14aa04865384fb90da250975/contracts/EIP5058/ERC5058.sol https://github.com/radiocaca/openpfp-contracts/blob/7423cd516965b188c617be5364bddd01c0d78088/contracts/OpenPFPExchange.sol	**Github**	N/A

13th June 2022

3. Audit Scope

ID	File	SHA-256 checksum
ERC721L/contracts/EIP5058	ERC5058.sol	754F975F93F0E6F858ECC1FD868CA42C1D672019EB122EC8A6F1E94DD6F5F673
ERC721L/contracts/EIP5058	IERC5058.sol	F248F94FD4786CB86599741BB30B05611491976230D0D5130A895AD6B5B80D24
ERC721L/contracts/factory	IERC5058Factory.sol	C53ECD67874D5F818898885EF13659F12C2548BDBE1FA40758507568F6EDF706
ERC721L/contracts/factory	IERC721Bound.sol	532FAA8E50205ACB0CFD09834D802B488DCDF7F815FFE79DC4E872C5EAF68959
ERC721L/contracts/EIP5058Upgradeable	ERC721Lockable.sol	926BB23FC1E0ECB2A5F65DAE7BDAC11C12A97E8C106E9A9B49B0328E94D29A2D
ERC721L/contracts/EIP5058Upgradeable	IERC721Lockable.sol	105FBB55549E603176C7152A4F2D6D8EAD4CF68BE5B49A913A14F2D91BC60A96
ERC721L/contracts/EIP5058/extensions	ERC5058Bound.sol	4F923113D2C8CC28FC2F89C77063EAA95BBD50895F6832F5CFDE593578DC487D
ERC721L/contracts/NFTs	MatrixPlusBox.sol	179FA3ECC43AF42CC08F20A8648AFA1A4DBE44A391C0FCDED25A888E6981DF3D
ERC721L/contracts/utils	ERC721Attachable.sol	CEAB4F2F35D3C8B2362159A8713C58D781E0DFC0A09EADA635FD955E64D2AC83
ERC721L/contracts/utils	TokenWithdraw.sol	FA562B879727A3E5E77A19BCC4F46FFA484B27E9165B942E7B4757A5A0935B02
openpfp-contracts/contracts	OpenPFPExchange.sol	A2A52D13F315735122E99CC38305D061A6910E3976A2C6EDDD8BEAD4F5B57CFD
openpfp-contracts/contracts	CurrencyManager.sol	CFEE8CA18CB1F11FEA0D71B74ACE9C0CB533D4501E7566005CB95FBF32F804BC
openpfp-contracts/contracts	ExecutionManager.sol	392360B9A0583677773F940CAFCD37DA85A91B42CA9ACF22EC28DB2174358E0A
openpfp-contracts/contracts	RoyaltyFeeManager.sol	AEDCF4C4ACF9DAA7FE954C501FB630275B04A5577CBB5E0C485D5854A9987857
openpfp-contracts/contracts	TransferSelectorNFT.sol	236E95E82FE35162B2F95122E51E26DA859ED763DD473F5EDC19713023CDC77C
openpfp-contracts/contracts/interfaces	ICurrencyManager.sol	2C56DB1882DF93873F6C0D6FC5E8B9BB060D0F167FA2E8996255E5F6360419D2

ID	File	SHA-256 checksum
openpfp-contracts/contracts/interfaces	IExecutionManager.sol	4F16AED94A69248306D307B0EBA26E355279C5367AB4E1B75F464F50B6B6F2FC
openpfp-contracts/contracts/interfaces	IExecutionStrategy.sol	3C331C764DCF6A2CB683CCAAB679D0669547D32502CAA0A16446BBB3824DC90A
openpfp-contracts/contracts/interfaces	IOpenPFPExchange.sol	237C4A31AE2E2AA93A116C30FEF3E29FEEF74A32A910D06717D86B263A6FBA2C
openpfp-contracts/contracts/interfaces	IRoyaltyFeeManager.sol	B9799D9FA53FD9A43EE819D3B37CEDBA93466D485F55816A707375635F7F8357
openpfp-contracts/contracts/interfaces	IRoyaltyFeeRegistry.sol	854CB3632E17DA1AD14EF32C344AFDCA4E9BF1A35541EF85907DF02883C45A9B
openpfp-contracts/contracts/interfaces	ITransferManagerNFT.sol	7A6A94CC3BA8ACF57E6BBA0EA1115E2B40327A8B8CFF799788AB3E37604F6509
openpfp-contracts/contracts/interfaces	ITransferSelectorNFT.sol	DC645A8C0CB6F3DE6597E88822026044CC1C973AA69900A4C1E09F3D686158EC
openpfp-contracts/contracts/interfaces	IWETH.sol	69007C90A776A8EFE8F5890BBAE1A6F9F8567A743603A99AF32B6087EC92E73D
openpfp-contracts/contracts/libraries	OrderTypes.sol	A2038E91D121922C2DA0A7B2F2EA104E6E64C662F944068F55AC731541C79F49
openpfp-contracts/contracts/libraries	SignatureChecker.sol	82A3FE05B58130C843CD2B4A8138D9E45694321BD118456595294B12AAB7BEFF
openpfp-contracts/contracts/royalty	RoyaltyFeeRegistry.sol	31DE9FAD57CDF308AE1EFD1691D140D69079EE82A055CBA05AB3F448598C6A09
openpfp-contracts/contracts/strategys	StrategyAnyItemFromCollectionForFixedPrice.sol	E730E946C7A680E0666FE5ACDF1A42DB77368E5127BD037ED853562E426BC969
openpfp-contracts/contracts/strategys	StrategyPrivateSale.sol	51D36BEEB4DE48383F43B57694D0FEF1C5FB43896193909F2F238F00ED121267
openpfp-contracts/contracts/strategys	StrategyStandardSaleForFixedPrice.sol	7172C3A854A570617814085B30C46C58083C1D98E0D97DA24E3AE03DFE46C157
openpfp-contracts/contracts/transfers	TransferManagerERC1155.sol	B0F2BEE1F35906307412C77E89E02AACE0CD11AD575E3EE02FC7283A79D59DFA
openpfp-contracts/contracts/transfers	TransferManagerERC721.sol	239F07AE00E5D58F64DE9522D381440F911E675D8D1D3C5C39A4F76188752422
openpfp-contracts/contracts/transfers	TransferManagerNonCompliantERC721.sol	B1B127F7C5CBAA723837F60F82C0397F6619FF34D29E5BE62F86AA36FC29EEF0

13th June 2022

4. Code Structure

```
# REC721L
 ├──EIP5058
 |    |   ERC5058.sol                    #NFT contract can be locked
 |    |   IERC5058.sol                   #Interface file
 |    ├──extensions
 |    |      ERC5058Bound.sol            #NFT extended bound token can be locked
 |    └──factory
 |           IERC5058Factory.sol         #Interface file
 |           IERC721Bound.sol            #Interface file
 ├──EIP5058Upgradeable
 |     ERC721Lockable.sol                #NFT contract can be locked
 |     IERC721Lockable.sol               #Interface file
 ├──NFTs
 |     MatrixPlusBox.sol                 #NFT extended token can be locked
 └──utils
      ERC721Attachable.sol               #Erc721 token extension
      TokenWithdraw.sol                  #Token transactions
```

13th June 2022

```
# openpfp-contracts
|   CurrencyManager.sol                                          #Currency management
|   ExecutionManager.sol                                         #Management of Policies
|   OpenPFPExchange.sol                                          #Pairing of orders on the order book
|   RoyaltyFeeManager.sol                                        #Management of privilege fees
|   TransferSelectorNFT.sol                                      #Management of privilege fees
|
├──interfaces                                                    #Interface file
|       ICurrencyManager.sol
|       IExecutionManager.sol
|       IExecutionStrategy.sol
|       IOpenPFPExchange.sol
|       IRoyaltyFeeManager.sol
|       IRoyaltyFeeRegistry.sol
|       ITransferManagerNFT.sol
|       ITransferSelectorNFT.sol
|       IWETH.sol
|
├──royalty
|       RoyaltyFeeRegistry.sol                                   #Registration of privilege fees
|
├──strategys
|       StrategyAnyItemFromCollectionForFixedPrice.sol           #Fixed price policy for any order
|       StrategyPrivateSale.sol                                  #Address specific policies
|       StrategyStandardSaleForFixedPrice.sol                    #Fixed price strategy for any buyer and seller
|
├──transfers
|       TransferManagerERC1155.sol                               #Transfer function based on ERC1155
|       TransferManagerERC721.sol                                #Transfer function based on ERC721
|       TransferManagerNonCompliantERC721.sol                    #Non secure transfer function based on ERC721
|
└──libraries
        OrderTypes.sol                                           #Format of order
        SignatureChecker.sol                                     #Signature check
```

13th June 2022

Audit Report Summary

1. Audit Methods

By clearly understanding the design purpose, operation principle and implementation mode of the project, the audit team conducted in-depth research and analysis of the contract code. Based on clarifying the calling relationship between each contract and its functions, the possible loopholes in the contract are located and analyzed. Finally, a document containing the problem descriptions and corresponding modification suggestions is formed.

Audit methods	Static analysis, Manual Review

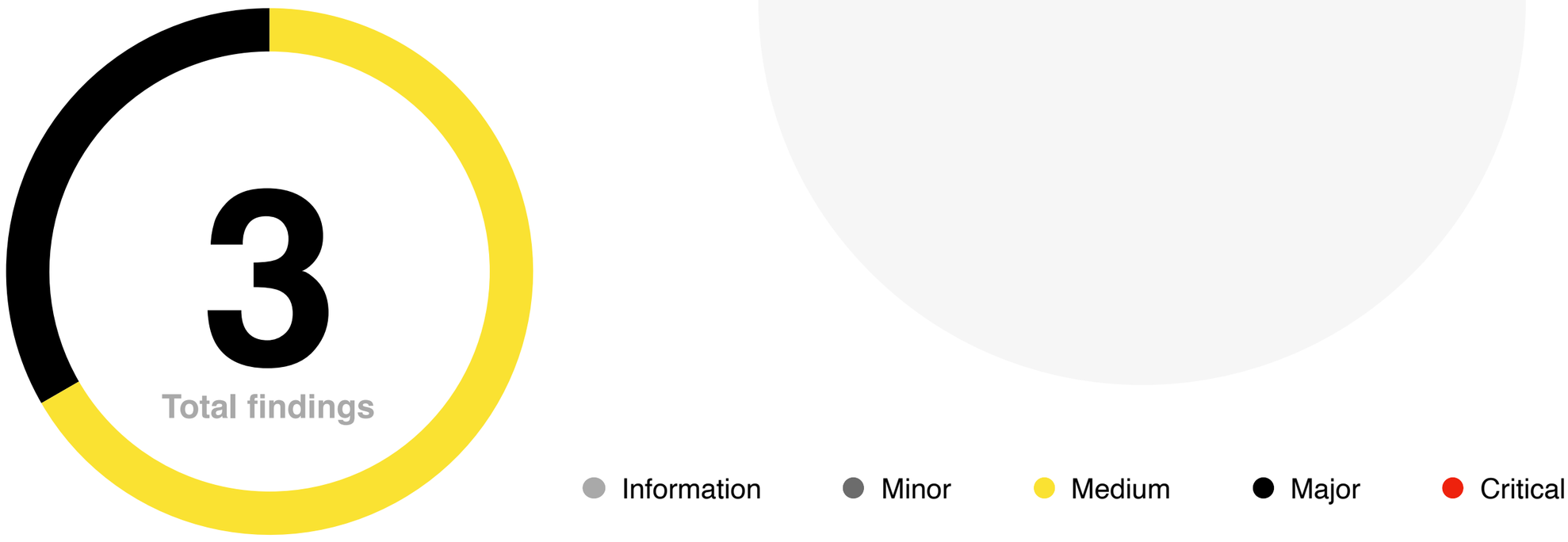
2. Audit Process

Steps	Operation	Description
1	Background	Reading the descriptions, white papers, contract source code, and other relevant information the project team provides to ensure a proper understanding of project functions.
2	Automated testing	Automated detection tools will be mainly used to scan the source code to find common potential vulnerabilities
3	Manual reveiw	The code will be thoroughly reviewed line by line by engineers to find potential vulnerabilities
4	Logical proofread	The engineer will compare the understanding of the code with the information provided by the project and check whether the code implementation is in line with the white paper information.
5	Test case	Including test case design, test scope analysis, symbolic execution, etc.
6	Optimization items	Review the project from the aspects of maintainability, security and operability according to the application scenarios, call methods and the latest research results

3. Risk Levels

Risk level	Issue description
Critical	Fatal risks and hazards that need to fixed immediately.
Major	Some high risks and hazards that will lead to related problems that must be solved
Medium	Some moderate risks and pitfalls may lead to potential risks that will eventually need to be addressed
Minor	There are low risks and hazards, mainly details of various types of mishandling or warning messages, which can be set aside for the time being
Information	Some parts can be optimized, such problems can be shelved, but it is recommended that the final solution

4. Audit Results



ID	Audit project	Risk level	Status
1	Reentrancy	None	
2	Injection	None	
3	Authentication bypass	None	
4	MEV Possibility	None	
5	Revert	None	
6	Race condition	None	
7	Insufficient Gas Griefing	None	
8	The major impact of flash loans	None	
9	Unreasonable economic model	None	
10	Predictable random numbers	None	
11	Voting rights management confusion	None	

ID	Audit project	Risk level	Status
12	Privacy leak	None	
13	Improper use of time on chain	None	
14	Improper codes in fallback function	None	
15	Improper identification	None	
16	Inappropriate opcode	None	
17	Inappropriate assembly	None	
18	Constructor irregularities	None	
19	Return value irregularity	None	
20	Event irregularity	None	
21	Keywords irregularity	None	
22	Not following ERC standards	Medium	**Acknowledged**
23	Irregularity of condition judgment	None	
24	Risk of liquidity drain	None	
25	Centralization Risk	None	
26	Logic change risk	None	
27	Integer overflow	None	
28	Improper function visiblity	None	
29	Improper initialization of variables	None	
30	Improper contract calls	None	
31	Variable irregularities	None	
32	Replay	None	
33	Write to Arbitrary Storage Location	None	
34	Honeypot logic	None	
35	Hash collision	None	
36	Improper call of external function	Major	**Acknowledged**

* In the above table, if the status column is **Acknowledged**, the audit team has informed the project owner of the vulnerability. Still, the project owner has not made any changes to the vulnerability or has not announced to the audit team the progress of the changes to the vulnerability. If the status column is **Resolved**, the project owner has changed the exposure, and the audit team has confirmed the changes.

5. Risk and Modification Program

The following section provides detailed information about the risk items learned after the audit, including the type of risk, risk level, location of the issue, description of the problem, recommendations for changes, and feedback from the project owner.

1. Not following ERC standards

Location	Contract file	Risk Status	Risk level
Line 86-95	MatrixPlusBox.sol	⚠️ Acknowledged	**Medium**

① Description

Matrixplusbox is a kind of lockable NFT. During NFT locking, the token owner cannot initiate transfer operations for tokens. However, there is no check whether NFT is locked when destroying tokens.

② Recommendation

Before destroying the NFT token, it is recommended to add an inspection phase to ensure that the token can be destroyed only when it is unlocked.

③ Code

```JavaScript
function burn(uint256 tokenId) external {
    //@OKLink Audit Description: Missing check if NFT is locked
    //@OKLink Audit Solution: Add a check phase to ensure that tokens in
unlocked status can be destroyed
    require(
        _isApprovedOrOwner(_msgSender(), tokenId) ||
            hasRole(BURNER_ROLE, _msgSender()) ||
            masterOf(tokenId) == _msgSender(),
        "ERC721: caller is not owner nor approved"
    );

    _burn(tokenId);
}
```

13th June 2022

2. Not following ERC standards

Location	Contract file	Risk Status	Risk level
Line 32-53	ERC5080Bound.sol	⚠ Acknowledged	**Medium**

① Description

#1：Bound call<_ Lock > (perform NFT token locking operation) to execute casting, and call <unlockfrom> (perform NFT token unlocking operation) to execute destruction. The destroy operation of bound only exists in<_ Aftertokenlock> function, which is displayed in <unlockfrom><_ Lock> and<_ Burn> is called to and is based on the erc5080 standard. However, even if the <unlockfrom> function is not called, the NFT token will return to the unlocked state after the locking time, and the bound corresponding to the NFT token is still not destroyed.

#2：The Function <islocked> that judges locking status : the token will also return to the unlocked status after the locking time.

② Recommendation

It is suggested that the development team reconsider the judgment logic of NFT token locking status or the destruction logic of bound, so as to ensure that bound only exists during NFT token locking and is destroyed when NFT token is unlocked.

③ Code#1

```JavaScript
function _afterTokenLock(
    address operator,
    address from,
    uint256 tokenId,
    uint256 expired
) internal virtual override {
    super._afterTokenLock(operator, from, tokenId, expired);

    if (bound != address(0)) {
        if (expired != 0) {
            // lock mint
            if (operator != address(0)) {
                IERC721Bound(bound).safeMint(msg.sender, tokenId, "");
            }
        } else {
            // unlock
            if (IERC721Bound(bound).exists(tokenId)) {
                IERC721Bound(bound).burn(tokenId);
            }
        }
    }
}
```

③ Code#2

```C++
function isLocked(uint256 tokenId) public view virtual override returns (bool)
{
    return lockedTokens[tokenId] > block.timestamp;
}
```

13th June 2022

3. Improper call of external function

Location	Contract file	Risk Status	Risk level
Line 204、275、348	OpenPFPExchange.sol	⚠️ Acknowledged	**Major**

① Description

The <matchaskwithtakerbidusingethandwith>, <matchaskwithtakerbid>, <matchbidwithtakerask> three functions all accept the customized "MakerOrder" as the parameter and do not detect the security and availability of its content. Users can arbitrarily construct the content of the structure, and there is a risk of calling external malicious contracts in subsequent execution.

The problems of the above three functions are similar. The following uses <matchaskwithtakerbid > as an example. The function calls the address passed in by the "makerAsk.strategy" parameter at <IExecutionStrategy(makerAsk.strategy).canExecuteTakerBid()>, which can be maliciously constructed by the caller to customize the logic and return value of the call.

② Recommendation

It is suggested that the judgment logic for the white list of <makerask.strategy> can be added to ensure that the target address is secure and trusted.

③ Code 1

```JavaScript
function matchAskWithTakerBidUsingETHAndWETH(
        OrderTypes.TakerOrder calldata takerBid,
        OrderTypes.MakerOrder calldata makerAsk
    ) external payable override nonReentrant

function matchAskWithTakerBid(
        OrderTypes.TakerOrder calldata takerBid,
        OrderTypes.MakerOrder calldata makerAsk
    ) external override nonReentrant

function matchBidWithTakerAsk(
        OrderTypes.TakerOrder calldata takerAsk,
        OrderTypes.MakerOrder calldata makerBid
    ) external override nonReentrant
```

13th June 2022

③ Code 2

```JavaScript
function matchAskWithTakerBid(
    OrderTypes.TakerOrder calldata takerBid,
    OrderTypes.MakerOrder calldata makerAsk
) external override nonReentrant {
    require(
        (makerAsk.isOrderAsk) && (!takerBid.isOrderAsk),
        "Order: Wrong sides"
    );
    require(
        msg.sender == takerBid.taker,
        "Order: Taker must be the sender"
    );

    // Check the maker ask order
    bytes32 askHash = makerAsk.hash();
    //@OKLink Audit: Check signature only
    _validateOrder(makerAsk, askHash);

    //@OKLink Audit Description: The 'makerAsk.strategy' address was called
without checking its security.
    //@OKLink Audit Solution: Add white list judgment to "makerask.strategy"
    (
        bool isExecutionValid,
        uint256 tokenId,
        uint256 amount
    ) = IExecutionStrategy(makerAsk.strategy).canExecuteTakerBid(
        takerBid,
        makerAsk
    );
```

13th June 2022

```
require(isExecutionValid, "Strategy: Execution invalid");

// Update maker ask order status to true (prevents replay)
_isUserOrderNonceExecutedOrCancelled[makerAsk.signer][
    makerAsk.nonce
] = true;

// Execution part 1/2
_transferFeesAndFunds(
    makerAsk.strategy,
    makerAsk.collection,
    tokenId,
    makerAsk.currency,
    msg.sender,
    makerAsk.signer,
    takerBid.price,
    makerAsk.minPercentageToAsk
);

// Execution part 2/2
_transferNonFungibleToken(
    makerAsk.collection,
    makerAsk.signer,
    takerBid.taker,
    tokenId,
    amount
);

emit TakerBid(
    askHash,
    makerAsk.nonce,
    takerBid.taker,
    makerAsk.signer,
    makerAsk.strategy,
    makerAsk.currency,
    makerAsk.collection,
    tokenId,
    amount,
    takerBid.price
);
```

13th June 2022

Disclaimer

i. This audit report focuses only on the types of audits identified in the final report issued. Other unknown security vulnerabilities are not part of this audit, and we do not accept responsibility for them.

ii. We shall only issue an audit report based on an attack or vulnerability that existed or occurred before the issuance of the audit report. We cannot determine the likely impact on the security posture of our projects for new attacks or vulnerabilities that may exist or occur in the future, and we are not responsible for them.

iii. The security audit analysis and other elements of our published audit report shall be based solely on documents and materials (including, but not limited to, contract codes) provided to us by the Project Party before the release of the audit report. Such documents and materials shall not be untrue, inaccurate, uninformative, altered, deleted, or concealed, and if the documents and materials provided by the Project Party are false, inaccurate, uninformative, changed, deleted or hidden, or if the documents and materials provided by the Project Party are untrue, inaccurate, uninformative, altered, deleted or concealed, or if the documents and materials provided by the Project Party are uninformative, uninformative, altered, deleted or hidden. If the records and information provided by the Project Party are untrue, inaccurate, uninformative, altered, deleted, or concealed, or if changes to such documents and information are made after the issuance of the audit report, we shall not be liable for any loss or adverse effect arising from any inconsistency between the reflected and actual conditions.

iv. The Project Parties are aware that our audit report is based on documents and information provided by the Project Parties and relies on the technology currently available. However, due to the technical limitations of any organization, there is a possibility that our audit report may not fully detect all risks. Our audit team encourages the project development team and any interested parties to conduct subsequent testing and audits of the project.

v. The project owner warrants that the project for which we are engaged to provide audit or testing services is legal, compliant, and does not violate applicable laws. The audit report is for the project owner's reference only, and the contents, manner of obtaining, use of, and any services or resources involved in the audit report shall not be relied upon for investment, tax, legal, regulatory, or advisory purposes of any kind, and we shall not be liable therefor. The Project Party shall not refer to, quote, display, or send the Audit Report in whole or in part to any third party without our prior written consent. The Project Party shall bear any loss or liability arising from that place. We assume no responsibility for any reliance on or use of the audit report for any purpose.

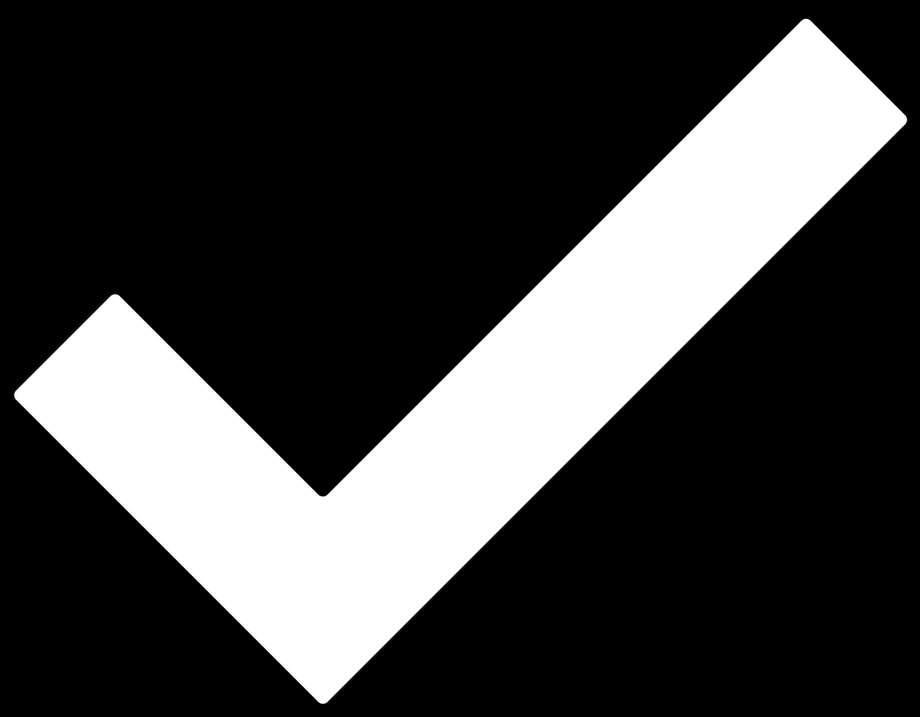
vi. This audit report does not cover the compiler of the contract or any areas beyond the programming language of the Smart Contract. The risk and liability of the audited Smart Contract arising from references to off-chain information or resources is the sole responsibility of the project party.

vii. Force Majeure. Force majeure means an unforeseen event whose occurrence and consequences cannot be avoided and cannot be overcome by the parties at the time of entering into the contract, including but not limited to natural disasters such as war, typhoon, flood, fire, earthquake, tidal wave, lightning, natural disaster, strike, nuclear explosion, epidemic and other unforeseen events such as changes in laws, regulations and policies and governmental acts, whose occurrence and consequences cannot be prevented or avoided, and which contains, affects or delays the performance by either party of all or part of its obligations under the contract.

viii. Suppose either party believes that the occurrence of force majeure affects the performance of its obligations under this Agreement. In that case, it shall promptly notify the other party and, depending on the extent of the effect of the event on the performance of the Agreement; the parties shall consult to determine whether to terminate the Agreement or partially relieve itself of its obligations to perform the Agreement, or to extend the performance of the Agreement.

ix. In force majeure, neither party shall be deemed in breach or non-performance of its obligations under this Agreement. Any financial commitments existing before the event shall not be affected, and the project party shall make payment for work performed by us.

13th June 2022



Passed.

Date	13th June 2022
Audit Team	歐科雲鏈

This audit covers two projects of ERC721L and openpfp-contracts written by Radiocaca based on the language of solidity. The focus is on the protocol's design, locking mechanism, and pre-transfer detection mechanism based on ERC721L standard token to find potential security risks. Review all aspects of problems in the project of openpfp-contracts, including pairing, cancellation, and placing orders in the form of order book, token transfer strategy, and rate setting, then discover potential security risks.