Competitive Security Assessment

ChaChaSwap

Sep 30th, 2022

Secure3

secure3.io

Summary

ChaChaSwap is the first creator-rights supportive decentralized multi-chain NFT AMM protocol. Its also supports royalty for the creator and innovative referral model helps community growth.

This report has been prepared for the project to identify issues and vulnerabilities in the smart contract source code. A comprehensive examination with Static Analysis and Manual Review techniques has been performed by Secure3 team. Also, a group of NDA covered experienced security experts have participated in the Secure3's Competitive Auditing as well to provide extra auditing coverage and scrutiny of the code.

The examination and auditing scope includes:

- Cross checking contract implementation against functionalities described in the documents and white paper disclosed by the project owner.
- Contract Privilege Role Review to provide more clarity on smart contract roles and privilege.
- Using static scanner to analyze smart contracts against common known vulnerabilities patterns.
- Verify the code base is compliant with the most up-to-date industry standards and best practices.
- Comprehensive line-by-line manual code review of the entire codebase by industry experts.

The security assessment resulted in findings that are categorized in three severity levels: Informational, Low, Medium, Critical. For each of the findings we have provided recommendation of a fix or mitigation for security and best practices.

Overview

Project Detail

Project Name	ChaChaSwap
Platform & Language	Ethereum, Solidity
Codebase	• https://github.com/chachaswap/smart-contract • audit commit - a426a8da7ef06fbebe1b20057fe51c891c735c05 • final commit - 52086d815db045d4efc4a3ff3f624e584b498c24
Audit Methodology	• Competitive Auditing • Business Logic and Code Review • Privileged Roles Review • Static Analysis

Code Vulnerability Review Summary

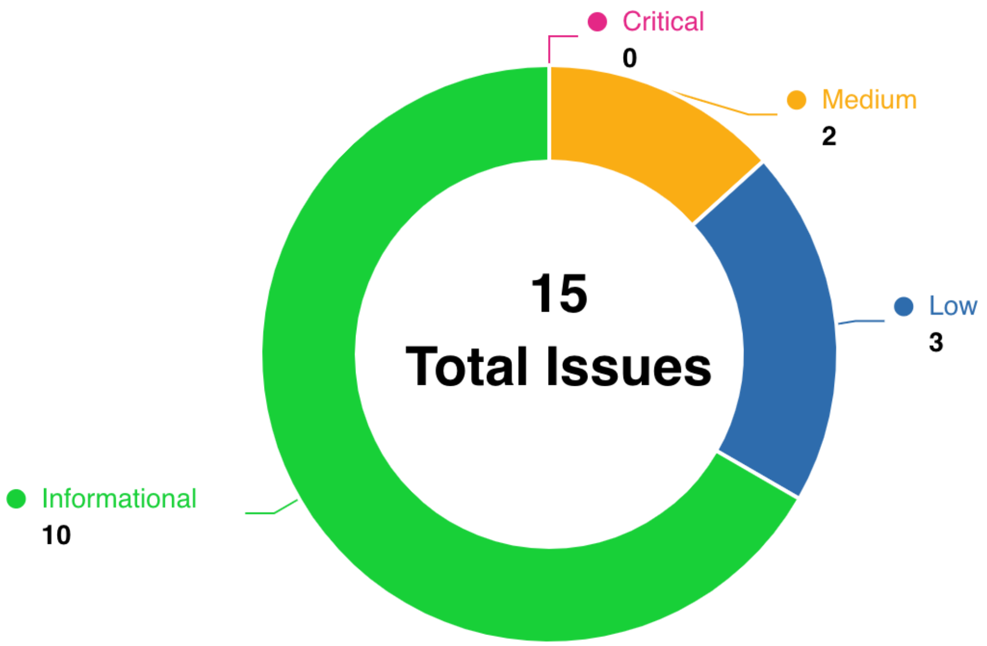
Vulnerability Level	Total	Reported	Acknowledged	Fixed	Mitigated	Declined
Critical	0	0	0	0	0	0
Medium	2	0	1	0	0	1
Low	3	0	0	3	0	0
Informational	10	0	3	6	1	0

Audit Scope

File	Commit Hash
contracts/LSSVMRouter.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/LSSVMPair.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/LSSVMPairFactory.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/rewardPool/ChachaPool.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/LSSVMPairERC20.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/royalty/RoyaltyManager.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/referralManager/ReferralManager.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/LSSVMPairETH.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/bonding-curves/ExponentialCurve.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/lib/LSSVMPairCloner.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/LSSVMPairMissingEnumerable.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/bonding-curves/LinearCurve.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/LSSVMPairEnumerable.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/swapRewardPool/SwapRewardPool.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/bonding-curves/ICurve.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/lib/OwnableWithTransferCallback.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/rewardPool/IChachaPool.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/referralManager/IReferralManager.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/ILSSVMPairFactoryLike.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/swapRewardPool/ISwapRewardPool.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/token/Vchacha.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/LSSVMPairMissingEnumerableERC20.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/LSSVMPairMissingEnumerableETH.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/lib/PermitAble.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/LSSVMPairEnumerableERC20.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/lib/ReentrancyGuard.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/LSSVMPairEnumerableETH.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/token/Chacha.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/bonding-curves/CurveErrorCodes.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/lib/IERC20Mintable.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05
contracts/royalty/IRoyaltyQualification.sol	a426a8da7ef06fbebe1b20057fe51c891c735c05

contracts/lib/IOwnershipTransferCallback.sol	**a426a8da7ef06fbebe1b20057fe51c891c735c05**
contracts/royalty/IRoyaltyManager.sol	**a426a8da7ef06fbebe1b20057fe51c891c735c05**

Code Assessment Findings



ID	Name	Category	Severity	Status	Contributor
CCS-1	`LSSVMPair` events are declared but not emitted	Code Style	Informational	Fixed	Hupixiong3
CCS-2	The owner can arbitrarily withdraw the deposit token	Privilege Related	Medium	Acknowledged	Hupixiong3, iczc, 0xoyst2r
CCS-3	`Pragma version ^0.8.0` allows old versions	Language Specific	Informational	Acknowledged	Hellobloc
CCS-4	`LSSVMPairETH` and `LSSVMPairERC20` same code snippets can be reused	Code Style	Informational	Fixed	iczc
CCS-5	Malicious inviter or royaltyReceiver will cause the transaction to be reverted	DOS	Low	Fixed	comcat
CCS-6	The function doesn't follow the `check-effect-interact` pattern	Code Style	Informational	Fixed	comcat
CCS-7	function `initialize` should check if `_owner` is valid	Logical	Informational	Fixed	Hellobloc
CCS-8	Missing events record for important functions	Language Specific	Informational	Mitigated	comcat, iczc,

					0xoyst2r, Hellobloc
CCS-9	Gas optimize by not using the SafeMath library	Gas Optimization	Informational	Fixed	comcat
CCS-10	Use new OpenZeppelin codebase version for improved security	Language Specific	Informational	Fixed	Hellobloc
CCS-11	Use ERC165Checker to check support interfaces	Logical	Low	Fixed	comcat, 0xoyst2r
CCS-12	`updateProjectConfig()` Incorrect permission control	Logical	Medium	Declined	iczc
CCS-13	`RoyaltyManager` should check that defaultReceiver is not `address(0)`	Logical	Low	Fixed	iczc
CCS-14	`SwapRewardPool` The same calculation result can be reused to save gas	Code Style	Informational	Acknowledged	iczc
CCS-15	constructor() parameters `preMintAddress` and `preMintAmount` are not used	Language Specific	Informational	Acknowledged	iczc

CCS-1: `LSSVMPair` events are declared but not emitted

Category	Severity	Code Reference	Status	Contributor
Code Style	**Informational**	**code\contracts\LSSVMPair.sol#189,260,322.**	**Fixed**	**Hupixiong3**

Code

```
    event SwapNFTInPair(address token, uint256 protocolFeeEarned, address inviter, uint256 inviterFeeBonus, address trader);
    event SwapNFTOutPair(address token, uint256 protocolFeeEarned, address inviter, uint256 inviterFeeBonus, address trader);

    event TokenDeposit(uint256 amount);

    event TokenWithdrawal(uint256 amount);
    event NFTWithdrawal();
```

Description

Hupixiong3 : Nonstandard use of event

Recommendation

Hupixiong3 : Add event as record

Client Response

The latest code uses events

CCS-2:The owner can arbitrarily withdraw the deposit token

Category	Severity	Code Reference	Status	Contributor
Privilege Related	Medium	code\contracts\rewardPool\ChachaPool.sol#412-414. contracts/rewardPool/ChachaPool.sol#L412	Acknowledged	Hupixiong3, iczc, 0xoyst2r

Code

```
    412:     function recovery(address _token, address _account, uint256 _amount) external
onlyOwner {
    413:         IERC20(_token).safeTransfer(_account, _amount);
    414:     }
```

Description

Hupixiong3 : The owner can arbitrarily retrieve the user's pledged assets through the recovery function
iczc : The `recovery()` function enables the owner to transfer the tokens deposited by the user in the contract, this gives the owner the privilege to transfer all assets of the contract.
0xoyst2r : The owner can transfer users' pledged token by the `recovery()`

Recommendation

Hupixiong3 : Transfer the deposit tokens to limit the scope between the contract balance and the actual pledge
iczc : `recovery()` should not allowed to transfer the chacha and vchacha token in the contract.

Client Response

The owner will use a MultiSig wallet

CCS-3: `Pragma version ^0.8.0` allows old versions

Category	Severity	Code Reference	Status	Contributor
Language Specific	Informational	code\contracts\LSSVMPair.sol#1045	Acknowledged	Hellobloc

Code

```
2:     pragma solidity ^0.8.0;

1045:  return abi.decode(_returnData, (string));
```

Description

Hellobloc : There are some bugs in the 0.8 series compiler due to feature updates. although no scenarios have been found that can be exploited, the code related to the above bug exists in ChaCha. For example:

- `abi.decode()` Bug

Recommendation

Hellobloc : In order to eliminate security risks, we recommend compiling the contract with the newer stable version (e.g. 0.8.16) of the compiler to eliminate possible problems.

Client Response

Adapted to 0.8.17, configured in hardhat.config.ts

CCS-4: `LSSVMPairETH` and `LSSVMPairERC20` same code snippets can be reused

Category	Severity	Code Reference	Status	Contributor
Code Style	Informational	contracts/LSSVMPairETH.sol#L53-L89, L150-L185 contracts/LSSVMPairETH.sol#L25,#L133 contracts/LSSVMPairERC20.sol#L39,#L227	Fixed	iczc

Code

```
//File: LSSVMPairETH.sol
53:     address inviter;
54:     uint256 bonus;
55:     if (isRouter) {
...

150:    address inviter;
151:    uint256 bonus;
152:    if (isRouter) {
...

//File: LSSVMPairERC20.sol
244:    address inviter;
245:    uint256 bonus;
246:    if (isRouter) {
...
```

Description

iczc : The invitation reward logic in `_pullTokenInputAndPayProtocolFee()` and `_payProtocolFeeFromPair()` of LSSVMPairETH contract is categorized into two cases that are called through a router or not, they have the same logic but still are implemented repeatedly.
iczc : `_pullTokenInputAndPayProtocolFee()` and `_payProtocolFeeFromPair()` of LSSVMPairETH and LSSVMPairERC20 have the same code for calculating invitation rewards except for the event and transfer.

Recommendation

iczc : Fetch the inviter first and reuse the reward logic.

```
address inviter;
if (isRouter) {
    inviter = IReferralManager(_factory.referralManager()).getInviter(routerCaller);
} else {
    inviter = IReferralManager(_factory.referralManager()).getInviter(msg.sender);
}
```

iczc : Recommend to implement common logic as private function and call it where required.

Client Response

Refactored

CCS-5:Malicious inviter or royaltyReceiver will cause the transaction to be reverted

Category	Severity	Code Reference	Status	Contributor
DOS	Low	contracts/LSSVMPairETH.sol#L72 contracts/LSSVMPairETH.sol#L118 contracts/LSSVMPairETH.sol#L168	Fixed	comcat

Code

```
71:     if (bonus > 0) {
72:         payable(address(inviter)).safeTransferETH(bonus);
73:     }

118:    payable(royaltyReceiver).safeTransferETH(royaltyAmount);

167:    if (bonus > 0) {
168:        payable(address(inviter)).safeTransferETH(bonus);
169:    }
```

Description

comcat : inside the `_pullTokenInputAndPayProtocolFee` function, it will transfer ETH to the refer's inviter. and also it use `solmate` `safeTransferETH` function, which inside is `address(to).call{value: value}("");`. so, there exists one possibility, that an attacker can revert all the tx, whose inviter has been set as the attacker.

the same for the royalty fee sending, if the owner of the NFT is malicious, any ETH transfer to the owner of NFT will be revert. so all the NFT traded inside the chachaswap will revert as long as it sets the royalty info. just like:

```
contract Attacker {
      receive() external payable {
            revert();
      }
}
```

Recommendation

comcat : wrap the ETH into WETH, and transfer WETH instead of ETH.

Client Response

Indeed, now send eth to inviter and royalty receiver. And if fails, send eth to factory

CCS-6:The function doesn't follow the `check-effect-interact` pattern

Category	Severity	Code Reference	Status	Contributor
Code Style	Informational	contracts/LSSVMPairMissingEnumerable.sol#L35-L36 contracts/LSSVMPairMissingEnumerable.sol#L56-L58	Fixed	comcat

Code

```
56:     _nft.safeTransferFrom(address(this), nftRecipient, nftIds[i]);
57:     // Remove from id set
58:     idSet.remove(nftIds[i]);

35:     _nft.safeTransferFrom(address(this), nftRecipient, nftId);
36:     idSet.remove(nftId);
```

Description

comcat : when sending out the nft, inside the `_sendSpecificNFTsToRecipient` and `_sendAnyNFTsToRecipient`, it doesn't follow the `check-effect-interact` pattern.

Recommendation

comcat : should remove id from idset first, then transfer nft out. since, ERC721 has a callback.

```
    function _sendAnyNFTsToRecipient(
        IERC721 _nft,
        address nftRecipient,
        uint256 numNFTs
    )
        internal
        override
        returns (uint256[] memory nftIds)
    {
        nftIds = new uint256[](numNFTs);
        // Send NFTs to recipient
        // We're missing enumerable, so we also update the pair's own ID set
        // NOTE: We start from last index to first index to save on gas
        uint256 lastIndex = idSet.length() - 1;
        for (uint256 i = 0; i < numNFTs;) {
            uint256 nftId = idSet.at(lastIndex);
            idSet.remove(nftId);
            nftIds[i] = nftId;
            unchecked {
                --lastIndex;
                ++i;
            }
            _nft.safeTransferFrom(address(this), nftRecipient, nftId);
        }
    }
    function _sendSpecificNFTsToRecipient(
        IERC721 _nft,
        address nftRecipient,
        uint256[] calldata nftIds
    )
        internal
        override
    {
        // Send NFTs to caller
        // If missing enumerable, update pool's own ID set
        uint256 numNFTs = nftIds.length;
        for (uint256 i; i < numNFTs;) {

            // Remove from id set
            idSet.remove(nftIds[i]);

            unchecked {
                ++i;
            }
           _nft.safeTransferFrom(address(this), nftRecipient, nftIds[i]);
```

```
            }
        }
```

Client Response

Accepted, but leave i++ still at the end of for loop

CCS-7:function `initialize` should check if `_owner` is valid

Category	Severity	Code Reference	Status	Contributor
Logical	Informational	code\contracts\LSSVMPair.sol#84-116	Fixed	Hellobloc

Code

```
84:     function initialize(
85:         address _owner,
86:         address payable _assetRecipient,
87:         uint128 _delta,
88:         uint96 _fee,
89:         uint128 _spotPrice
90:     ) external payable {
91:         require(owner() == address(0), "Initialized");
92:         __Ownable_init(_owner);
...
```

Description

Hellobloc : The current `initialize` lock implementation relies on a determination of whether the owner is `0`, but given that users can create their own Pair and this Pair can pass the verification of Router and Fatory.. This allows the user to set the owner to `0x0` and perform double `initialize`, thus preventing emit events while updating `_owner`, `_assetRecipient`, `_delta`, `_fee` and `_spotPrice`.

Recommendation

Hellobloc : We recommend checking that `_owner` is not zero address in the initialize of LSSVMPair to ensure that LSSVMPair can only be initialized once and emit events in the initialize phase. For example.

```
function initialize(
        address _owner,
        address payable _assetRecipient,
        uint128 _delta,
        uint96 _fee,
        uint128 _spotPrice
    ) external payable {
        require(owner() == address(0), "Initialized");
        require(_owner != address(0), "Owner Cannot Be 0x0");
        ...
        emit SpotPriceUpdate(_spotPrice);
        emit DeltaUpdate(_delta);
        emit AssetRecipientChange(_assetRecipient);
        emit FeeUpdate(_fee);
    }
```

Client Response

Accepted

CCS-8:Missing events record for important functions

Category	Severity	Code Reference	Status	Contributor
Language Specific	Informational	code\contracts\lib\PermitAble.sol#20-24 contracts/swapRewardPool/SwapRewardPool.sol#L25 contracts/referralManager/ReferralManager.sol#L42-L73 contracts/royalty/RoyaltyManager.sol#L59-L63	Mitigated	comcat, iczc, 0xoyst2r, Hellobloc

Code

```
// File: ReferralManager.sol
42:     function _register(address account, bytes32 referralCode) internal {

// File: RoyaltyManager.sol
59:     function initializeManagerConfig(

// File: PermitAble.sol
20:     function setPermission(address[] calldata who, bool[] calldata be) external onlyOwner {

// File: SwapRewardPool.sol
25:     function createAirDrop20(AirDrop20Config memory param) override external onlyOwner {
```

Description

comcat : the referral system requires user to register in, and when user call the `register(bytes32)` or the owner call the `register(address,bytes32)`, there is not event get emited. which is not good for monitor.
comcat : inside the RoyaltyManger contract, all the configure function lack the corresponding event. it is better to emit event for the configure functions,as follows. especially for an NFT trading platform, it is very important to emit the corresponding event, so that you can setup a monitor system, to help monitor the project.

```
initializeManagerConfig,
initializeProjectConfig,
updateManagerConfig,
updateProjectConfig,
setManagerSpecific,
setProjectSpecific,
setMaxFeeRateLimitPer10000
```

iczc : `createAirDrop20()` is used to create a new reward pool, the function does not emit an event.
Hellobloc : Currently, some important functions in the `ChaCha` contract lack `event`, which may result in important data updates in the contract not being synchronized to off-chain in a timely manner. For example: **Lack of Event**

- `setPermission` in code\contracts\lib\PermitAble.sol

- `_updateVest` `updateRewardRate` `constructor` `recovery` in code\contracts\rewardPool\ChachaPool.sol
- `createAirDrop20` in code\contracts\swapRewardPool\SwapRewardPool.sol
- `initializeManagerConfig` `initializeProjectConfig` `updateManagerConfig` `updateProjectConfig` `changeProjectConfigOperator` `setManagerSpecific` `setProjectSpecific` and `setMaxFeeRateLimitPer10000in` in code\contracts\royalty\RoyaltyManager.sol

Lack of event content

- `createPairETH` in code\contracts\LSSVMPairFactory.sol ...

Recommendation

comcat : emit event when user register in

```
event Register(address indexed user,bytes32 indexed referralCode);
function _register(address account, bytes32 referralCode) internal {
     emit Register(account, referralCode);
}
```

comcat : you may define Events like:

```
    event ManagerConfigInit(address indexed nftAddress,address indexed defaultReceiver,uint256
defaultFeeRatePer10000);
    event ProjectConfigInit(address indexed nftAddress,address indexed defaultReceiver,uint256
defaultFeeRatePer10000);
    event ManagerConfigUpdated(address indexed nftAddress,address indexed defaultReceiver,uint256
defaultFeeRatePer10000);
    event ProjectConfigUpdated(address indexed nftAddress,address indexed defaultReceiver,uint256
defaultFeeRatePer10000);
    event OperatorChanged(address indexed nftAddress,address indexed oldOperator, address indexed
newOperator);
    event ManagerSpecific(address indexed nftAddress,uint256 indexed nftId,bool enabled,address
receiver,uint256 feeRatePer10000);
    event MaxFeeRateLimitPer10000(uint256 indexed oldLimit,uint256 indexed newLimit);
```

and emit the corresponding event inside the funciton.

iczc : Emit an event with erc20Address at the end of `createAirDrop20()`.

0xoyst2r : Emit event at the end of the functions

Hellobloc : We recommend refinements to the event mechanism, specifically.

1. Emit events for important operations
2. Event content should include important content as much as possible
3. Events of different operations should not collide.

Client Response

Some of advise is needed, some are not. Added needed events

CCS-9:Gas optimize by not using the SafeMath library

Category	Severity	Code Reference	Status	Contributor
Gas Optimization	Informational	contracts/referralManager/ReferralManager.sol#L14	Fixed	comcat

Code

```
14:     using SafeMath for uint256;
```

Description

comcat : since the project's solidity version is already ^0.8, it is no need to use safe math for the math calculation.

Recommendation

comcat : remove the safe math library. use + instead of add, etc. which can save some gas

Client Response

Mis-import and mis-using, removed

CCS-10:Use new OpenZeppelin codebase version for improved security

Category	Severity	Code Reference	Status	Contributor
Language Specific	**Informational**	**code\contracts\royalty\RoyaltyManager.sol#70-71**	**Fixed**	**Hellobloc**

Code

```
// File: ERC165Checker.sol
1:     // SPDX-License-Identifier: MIT
2:     // OpenZeppelin Contracts v4.4.1 (utils/introspection/ERC165Checker.sol)

// File: RoyaltyManager.sol
68:     require(
69:         IERC165(nftAddress).supportsInterface(INTERFACE_ID_ERC721) ||
70:         IERC165(nftAddress).supportsInterface(INTERFACE_ID_ERC1155),
71:         "neither ERC721 nor ERC1155"
72:     );
```

Description

Hellobloc : Two vulnerabilities related to the `ERC165Checker` code base have been published by `openzeppelin`. Although no scenarios have been found for the `ChaCha` contract to be exploited for now, this could raise the security risk of `ChaCha` contract.

- ERC165Checker unbounded gas consumption
- ERC165Checker may revert instead of returning false

Recommendation

Hellobloc : We recommend using the latest `openzeppelin` library to eliminate the possibility of the above vulnerabilities.

Client Response

Upgrade OpenZeppelin to latest, which is build 4.7.3

CCS-11:Use ERC165Checker to check support interfaces

Category	Severity	Code Reference	Status	Contributor
Logical	Low	contracts/royalty/RoyaltyManager.sol#L69-L71 contracts/royalty/RoyaltyManager.sol#L93-L96	Fixed	comcat, 0xoyst2r

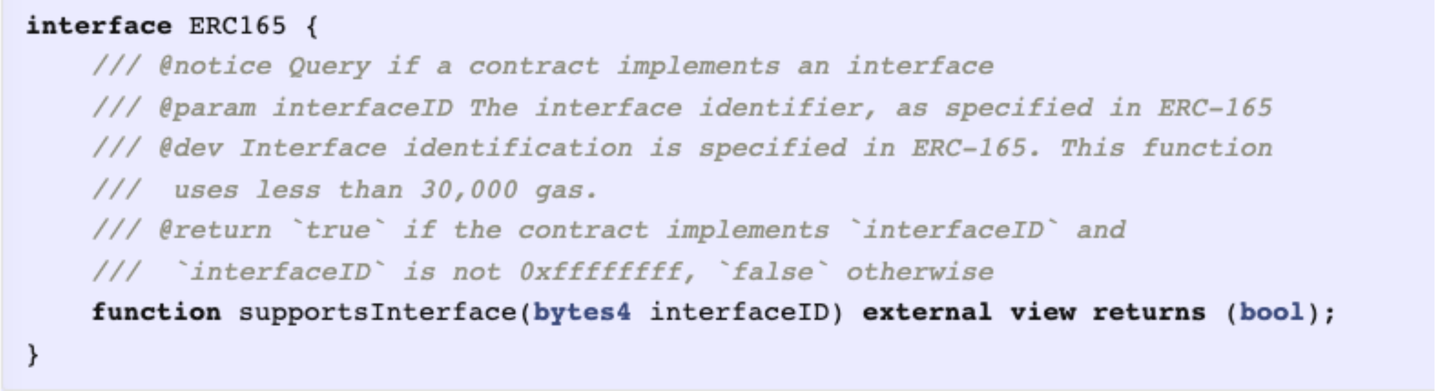
Code

```
68:     require(
69:         IERC165(nftAddress).supportsInterface(INTERFACE_ID_ERC721) ||
70:         IERC165(nftAddress).supportsInterface(INTERFACE_ID_ERC1155),
71:         "neither ERC721 nor ERC1155"
72:     );

93:     require(
94:         IERC165(nftAddress).supportsInterface(INTERFACE_ID_ERC721) ||
95:         IERC165(nftAddress).supportsInterface(INTERFACE_ID_ERC1155),
96:         "neither ERC721 nor ERC1155"
97:     );
```

Description

comcat : Make it compliance to ERC165 standard when judge ERC721/ERC1155. inside the RoyaltyManager contract's `initializeManagerConfig` function, it will judge whether an nft address is ERC721 or ERC1155. however, the way it use is just query the `IERC165(nftAddress).supportsInterface(INTERFACE_ID_ERC721)` which is not sufficient according to the EIP-721 standard.



```
interface ERC165 {
    /// @notice Query if a contract implements an interface
    /// @param interfaceID The interface identifier, as specified in ERC-165
    /// @dev Interface identification is specified in ERC-165. This function
    ///  uses less than 30,000 gas.
    /// @return `true` if the contract implements `interfaceID` and
    ///  `interfaceID` is not 0xffffffff, `false` otherwise
    function supportsInterface(bytes4 interfaceID) external view returns (bool);
}
```

0xoyst2r : use ERC165 standard to check if ERC721 or ERC1155 is supported

Recommendation

comcat : use the Openzeppelin's library:

```
 import {ERC165Checker} from
    "./dependency/openzeppelin-contracts/contracts/utils/introspection/ERC165Checker.sol";
```

```
  require(
            ERC165Checker.supportsInterface(nftAddress,INTERFACE_ID_ERC721) ||
            ERC165Checker.supportsInterface(nftAddress,INTERFACE_ID_ERC1155),
            "neither ERC721 nor ERC1155"
        );
```

0xoyst2r : use OpenZeppelin checker for introspection - https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/introspection/ERC165Checker.sol

Client Response

Accepted 165Checker

CCS-12: `updateProjectConfig()` Incorrect permission control

Category	Severity	Code Reference	Status	Contributor
Logical	**Medium**	**contracts/royalty/RoyaltyManager.sol #L131**	**Declined**	**iczc**

Code

```
127:     function updateManagerConfig(
128:         address nftAddress,
129:         address defaultReceiver,
130:         uint256 defaultFeeRatePer10000
131:     ) external onlyOwner {
```

Description

iczc : ProjectConfig is designed to permit the NFT owner to configure, but the `updateProjectConfig()` function was accidentally added the onlyOwner modifier. Therefore the condition that sender is contract owner and NFT owner is unlikely to be achieved, and this leads to ProjectConfig will not be able to updated.

Recommendation

iczc : Remove the onlyOwner modifier of `updateProjectConfig()`.

Client Response

Work as design, the ManagerConfig is under full control of 'Manager'

CCS-13: `RoyaltyManager` should check that defaultReceiver is not `address(0)`

Category	Severity	Code Reference	Status	Contributor
Logical	Low	contracts/royalty/RoyaltyManager.sol #L59,L82,L127,L140	Fixed	iczc

Code

```
59:     function initializeManagerConfig(
82:     function initializeProjectConfig(
127:    function updateManagerConfig(
140:    function updateProjectConfig(

77:     feeSetting.defaultReceiver = defaultReceiver;
```

Description

iczc : There is no check to defaultReceiver not zero-address in `initializeManagerConfig()`, `initializeProjectConfig()`, `updateManagerConfig()` and `updateProjectConfig()` of RoyaltyManager, This leads to the possibility of royalties accidentally set to black hole address.

Recommendation

iczc : Add `require(defaultReceiver != address(0))` statement.

Client Response

Accepted, but in the design, the receiver could be 0x00 to achieve 'burn' token.

CCS-14: `SwapRewardPool` The same calculation result can be reused to save gas

Category	Severity	Code Reference	Status	Contributor
Code Style	Informational	contracts/swapRewardPool/SwapRewardPool.sol#L44-L45	Acknowledged	iczc

Code

```
    58:     function claim20(AirDrop20Proof memory request) override external {
    59:
    60:         require(request.who == msg.sender, "you can not claim for other's");
    61:
    62:         require(claimable20(request) == 0, "not claimable");
    63:
    64:         bytes32 root = request.proof[request.proof.length - 1];
    65:         bytes32 leaf = keccak256(abi.encode(request.who, request.amount));
    ...

    39:     function claimable20(AirDrop20Proof memory request) override view public returns
(uint256){
    40:         if (request.proof.length < 2) {
    41:             return 1;
    42:         }
    43:
    44:         bytes32 root = request.proof[request.proof.length - 1];
    45:         bytes32 leaf = keccak256(abi.encode(request.who, request.amount));
    ...
```

Description

iczc : The process of computing leaf node exists in both the `claimable20()` and `claim20()`, this brings code redundancy and unnecessary overhead.

Recommendation

iczc : Calculate the hash once in `claim20()`, and then pass it to `claimable20()` via the parameter.

```
function claimable20(bytes32 memory root, bytes32 memory leaf) override view public returns
(uint256)
```

Client Response

Work as desgin, the view function is only for website to get message and the error goes independant

CCS-15:constructor() parameters `preMintAddress` and `preMintAmount` are not used

Category	Severity	Code Reference	Status	Contributor
Language Specific	Informational	contracts/token/Chacha.sol#L10-L11 contracts/token/Vchacha.sol#L10-L11	Acknowledged	iczc

Code

```
// File: Chacha.sol
08: contract Chacha is ERC20Capped, Ownable {
09:     constructor(
10:         address[] memory /*preMintAddress*/,
11:         uint256[] memory /*preMintAmount*/
12:     )
...

// File: Vchacha.sol
10:     constructor(
11:         address[] memory preMintAddress,
12:         uint256[] memory preMintAmount
13:     )
14:     ERC20("VChacha", "VChacha"){
15:     }
```

Description

iczc : The constructor parameter of the Chacha contract is unused.
iczc : The constructor parameter of the VChacha contract is unused.

Recommendation

iczc : Remove the parameter.
iczc : Remove the parameter.

Client Response

The given contract is for test, now the premint take effects.

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 Invoices, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Invoice. This report provided in connection with the services set forth in the Invoices shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Invoice. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Secure3’s prior written consent in each instance.

This report is not an “endorsement” or “disapproval” of any particular project or team. This report is not an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Secure3 to perform a security assessment. This report does not provide any warranty or guarantee of free of bug of codes analyzed, nor do they provide any indication of the technologies, business model or legal compliancy.

This report should not be used in any way to make decisions around investment or involvement with any particular project. Instead, it represents an extensive assessing process intending to help our customers increase the quality of their code and high-level consistency of implementation and business model, while reducing the risk presented by cryptographic tokens and blockchain technology.

Secure3’s position on the final decisions over blockchain technologies and corresponding associated transactions is that each company and individual are responsible for their own due diligence and continuous security.

The assessment services provided by Secure3 is subject to dependencies and under continuing development. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.