THE RANCH

Smart Contract Security Audit

Prepared by ShellBoxes

July 25th, 2022 - August 10th, 2022

Shellboxes.com

contact@shellboxes.com

Document Properties

Client	DefiBulls
Version	1.0
Classification	Public

Scope

The THE RANCH Contract in the THE RANCH Repository

Repo	Commit Hash
https://github.com/defibulls/TheRanchBTCBulls	37f52347c0980fe539942a5e3906912a1817ebed
https://github.com/defibulls/TheRanchBTCBulls_AUDITFINAL	6bccf85e3aa6b8cb0e0165212856a9a7de964327

Files	MD5 Hash
TheRanchBTCBullsCommunity.sol	37a18e98c8c8edc3b0a29c5c4349ae17
transparent_proxy/TRBCProxy.sol	53fab80036d5f9cd5f1a043d601a333b

Re-Audit Files

Files	MD5 Hash
transparent_proxy/TRBCProxy.sol	53fab80036d5f9cd5f1a043d601a333b
TheRanchBTCBullsCommunity.sol	b342e2aa3afe66949b1401e230a7264d

Contacts

COMPANY	EMAIL
ShellBoxes	contact@shellboxes.com

Contents

1 Introduction

DefiBulls engaged ShellBoxes to conduct a security assessment on the THE RANCH beginning on July 25th, 2022 and ending August 10th, 2022. In this report, we detail our methodical approach to evaluate potential security issues associated with the implementation of smart contracts, by exposing possible semantic discrepancies between the smart contract code and design document, and by recommending additional ideas to optimize the existing code. Our findings indicate that the current version of smart contracts can still be enhanced further due to the presence of many security and performance concerns.

This document summarizes the findings of our audit.

1.1 About DefiBulls

The Ranch is a defi ecosystem that is being built on Polygon network. TheRanch aims to provide several Dapps that benefit each other and reward early adopters of the system. The goal of the platform is to provide NFT holders with passive income as we invest into sustainable projects and start becoming validators and delegators firstly on the polygon blockchain then move to other blockchains.

Issuer	DefiBulls
Website	`https://theranch.community`
Type	Solidity Smart Contract
Audit Method	Whitebox

1.2 Approach & Methodology

ShellBoxes used a combination of manual and automated security testing to achieve a balance between efficiency, timeliness, practicability, and correctness within the audit's scope. While manual testing is advised for identifying problems in logic, procedure, and implementation, automated testing techniques help to expand the coverage of smart contracts and can quickly detect code that does not comply with security best practices.

1.2.1 Risk Methodology

Vulnerabilities or bugs identified by ShellBoxes are ranked using a risk assessment technique that considers both the LIKELIHOOD and IMPACT of a security incident. This framework is effective at conveying the features and consequences of technological vulnerabilities.

Its quantitative paradigm enables repeatable and precise measurement, while also revealing the underlying susceptibility characteristics that were used to calculate the Risk scores. A risk level will be assigned to each vulnerability on a scale of 5 to 1, with 5 indicating the greatest possibility or impact.

- Likelihood quantifies the probability of a certain vulnerability being discovered and exploited in the untamed.
- Impact quantifies the technical and economic costs of a successful attack.
- Severity indicates the risk's overall criticality.

Probability and impact are classified into three categories: H, M, and L, which correspond to high, medium, and low, respectively. Severity is determined by probability and impact and is categorized into four levels, namely Critical, High, Medium, and Low.

Impact	Likelihood: High	Likelihood: Medium	Likelihood: Low
High	Critical	High	Medium
Medium	High	Medium	Low
Low	Medium	Low	Low

2 Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the THE RANCH implementation. During the first part of our audit, we examine the smart contract source code and run the codebase via a static code analyzer. The objective here is to find known coding problems statically and then manually check (reject or confirm) issues highlighted by the tool. Additionally, we check business logics, system processes, and DeFi-related components manually to identify potential hazards and/or defects.

2.2 Key Findings

In general, these smart contracts are well-designed and constructed, but their implementation might be improved by addressing the discovered flaws, which include 1 critical-severity, 3 high-severity, 3 medium-severity, 4 low-severity vulnerabilities.

Vulnerabilities	Severity	Status
A.1. The Ecosystem Contracts Can Drain The TheRanchBTCBullsCommunity Contract	CRITICAL	Fixed
A.2. Rounding Errors Can Lead To Unexpected Values	HIGH	Fixed
A.3. mintingRaffle Randomness Can Be Controlled	HIGH	Acknowledged
A.4. The Owner Can Withdraw All The Tokens	HIGH	Fixed
A.5. The User Can Add Any Address As A Partner	MEDIUM	Mitigated
A.6. Race Condition	MEDIUM	Fixed
A.7. Centralization Risk	MEDIUM	Acknowledged
A.8. Avoid using .transfer() to transfer Ether	LOW	Fixed
A.9. For Loop Over Dynamic Array	LOW	Acknowledged
A.10. Owner Can Renounce Ownership	LOW	Fixed
A.11. Floating Pragma	LOW	Fixed

3 Finding Details

A TheRanchBTCBullsCommunity.sol

A.1 The Ecosystem Contracts Can Drain The TheRanchBTCBullsCommunity Contract [CRITICAL]

Description:

The ecosystem contracts update the USDC amount for the BTC Bulls owners on the TheRanchBTCBullsCommunity contract using the updateUsdcBonusFromAnother-Contract function. The ecosystem contract, however, does not provide the necessary USDC funding for the TheRanchBTCBullsCommunity contract. Therefore, the ecosystem contract can drain the balance of the TheRanchBTCBullsCommunity contract.

Code:

Listing 1: TheRanchBTCBullsCommunity.sol

```
490 function updateUsdcBonusFromAnotherContract(address[] memory
        ↪ _ownersOfTheNFTs, uint256 _amountToAdd) external {
491     require(isEcosystemRole[msg.sender] == true, "must be approved to
            ↪ interact");
```

Risk Level:

Likelihood – 5
Impact - 5

Recommendation:

The _amountToAdd * _ownersOfTheNFTs.length of USDC tokens should be approved by the ecosystem contract in order to fund the TheRanchBTCBullsCommunity contract.
You will also need to consider adding a safeTransferFrom call that will reclaim the approved tokens in the updateUsdcBonusFromAnotherContract.

Status - Fixed

The DefiBulls team has fixed the issue by requiring the ecosystem contract to fund the TheRanchBTCBullsCommunity contract with the needed amount using a safeTransferFrom call.

A.2 Rounding Errors Can Lead To Unexpected Values [HIGH]

Description:

The owner can set the payPerNftForTheMonth and currentRewardingDate variables using the setPayPerNftForTheMonthAndCurrentRewardingDate, these variables are used afterwards to distribute rewards using the rewardBulls function. This function contains multiple division operation which can lead to rounding errors. For any _totalAmountToDeposit that is lower than 13, the coreTeam_1_amt will be round to zero, and the same goes for coreTeam_2_amt for any value of _totalAmountToDeposit that is lower than 50. the _totalAmountToDeposit is used to get the value of the _disperableAmount, the payPerNftForTheMonth and the referralAmt variables.

Code:

Listing 2: TheRanchBTCBullsCommunity.sol

```
340 function setPayPerNftForTheMonthAndCurrentRewardingDate(uint256
        ↪ _totalAmountToDeposit, uint _dateOfRewarding) public onlyOwner {
341     if (lastDeposit != 0) { revert Rewarding_HasAlreadyHappenedThisMonth
            ↪ ();}
342     //if (_dateOfRewarding == currentRewardingDate) { revert
            ↪ Rewarding_HasAlreadyHappenedThisMonth();}


345     IERC20Upgradeable tokenContract = IERC20Upgradeable(
            ↪ wbtcTokenContract);
346     tokenContract.safeTransferFrom(msg.sender, address(this),
            ↪ _totalAmountToDeposit);

348     currentRewardingDate = _dateOfRewarding;
```

```
349     lastDeposit = _totalAmountToDeposit;


352     // in this function, lets pay out the core team first and then the
            ↪ 90% left gets divided up.
353     uint256 coreTeam_1_amt = _totalAmountToDeposit * 8 / 100;
354     uint256 coreTeam_2_amt = _totalAmountToDeposit * 2 / 100;

356     uint256 _disperableAmount = (_totalAmountToDeposit * 90 / 100);
357     uint256 payout_per_nft = _disperableAmount / _tokenSupply.current();
358     payPerNftForTheMonth = payout_per_nft;

360     btcBullOwners[coreTeam_1].WBTC_Balance += coreTeam_1_amt;
361     btcBullOwners[coreTeam_2].WBTC_Balance += coreTeam_2_amt;

363     // emit event
364     emit setPayPerNFTEvent(_totalAmountToDeposit, payout_per_nft,
            ↪ _dateOfRewarding);


367 }
```

Risk Level:

Likelihood – 4
Impact - 5

Recommendation:

To avoid all the rounding errors, the `_totalAmountToDeposit` should be verified to be higher or equal to 1 111 112.

Status - Fixed

The DefiBulls team has fixed the issue by requiring the `_totalAmountToDeposit` argument to be higher or equal to 1200000 in order to avoid a rounding error.

A.3 mintingRaffle Randomness Can Be Controlled [HIGH]

Description:

The mintingRaffle function is called by The ChainLinkVRF contract to reward randomly one of the daily raffle players. The TheRanchBTCBullsCommunity contract verifies that the calling contract has the ChainlinkVRF role. However, the owner has the ability to include any contract in the ChainlinkVRF role, therefore having the ability to select a specific winner.

Code:

Listing 3: TheRanchBTCBullsCommunity.sol

```
621 function mintingRaffle(uint _winningIndex) external {

623     require(isChainLinkVRFRole[msg.sender] == true, "must be approved to
        ↪  interact");

625     if (paused == false) { revert Pause_MustBePaused();}

627     address dailyRaffleWinner = dailyRafflePlayers[_winningIndex];
628     uint256 raffleWinningAmount = dailyRaffleBalance;

630     // update the daily raffle winnners USDC balance
631     btcBullOwners[dailyRaffleWinner].USDC_Balance += dailyRaffleBalance;

633     resetUserInDailyRaffle(); // must do before resetting
        ↪ dailyRafflePlayers
634     dailyRaffleBalance = 0; // reset dailyRaffleBalance back to zero
        ↪ after drawing
635     dailyRafflePlayers = new address[](0);

637     emit mintingRaffleEvent(_winningIndex, dailyRaffleWinner,
        ↪ raffleWinningAmount);}
```

Risk Level:

Likelihood – 4
Impact - 5

Recommendation:

Consider storing the bytecode's hash of the TheRanchBTCBullsChainLinkVRF contract, then verifying in the mintingRaffle that the sender's bytecode hash is the same as the one stored in the TheRanchBTCBullsCommunity contract.

Status - Acknowledged

The DefiBulls team has acknowledged the risk, stating that the community can directly verify the code of the ChainLinkVRFContract before the mint.

A.4 The Owner Can Withdraw All The Tokens [HIGH]

Description:

The owner can withdraw any number of tokens from the contract to his wallet using the withdrawToken method without any limitations. Therefore, the users who have yet to withdraw their balance along with hostingSafe, could face a denial of service, and can no longer withdraw their funds.

Code:

Listing 4: TheRanchBTCBullsCommunity.sol

```
679 function withdrawToken(address _tokenContract, uint256 _amount) external
      ↪ onlyOwner {
680     IERC20Upgradeable tokenContract = IERC20Upgradeable(_tokenContract);
681     tokenContract.safeTransfer(msg.sender, _amount);
682 }
```

Risk Level:

Likelihood – 4
Impact - 5

Recommendation:

It is recommended to limit the amount that can be withdrawn to be lower than the contract's balance - (btcMinersSafeBalance + hostingSafeBalance + USDCRewardsBalance) for the USDC token, and lower than the total of btcBullOwners.WBTC_Balance for the WBTC token.

Status - Fixed

The DefiBulls team has fixed the issue by limiting the withdrawn amount to tokenContract.balanceOf(address(this)) - (btcMinersSafeBalance + hostingSafeBalance + USDCRewardsBalance) for the USDC token, and to btcBullOwners[msg.sender].--WBTC_Balance in the case of WBTC token.

A.5 The User Can Add Any Address As A Partner [MEDIUM]

Description:

The contract has a referral system, when the mint, rewardBulls or updateUsdcBonusFromAnotherContract function is called, the referral system makes sure to reward the caller's partner. However, the user is able to add any address to be his partner using the setPartnerAddress function, therefore he can add an address of another wallet that he owns which will allow him to get the referralAmt. In addition to that, it allows a user that does not have a partner to be able to pay only referralAmt to a random partner instead of paying referralAmt*2 to the core team.

Code:

Listing 5: TheRanchBTCBullsCommunity.sol

```
655 function setPartnerAddress(address _newPartner) public {
656     if (address(_newPartner) == address(0)) { revert Partner_NotAllowed
        ↪ ();}
```

```
657     if (address(_newPartner) == msg.sender) { revert Partner_NotAllowed
            ↪ ();}

659     address currentPartner = myPartner[msg.sender];
660     // myPartner[msg.sender] = _newPartner;

662     if (currentPartner == address(0)){
663         myPartner[msg.sender] = _newPartner;
664         myParnterNetworkTeamCount[_newPartner] += 1;
665     } else {
666         myPartner[msg.sender] = _newPartner;
667         myParnterNetworkTeamCount[currentPartner] -= 1;
668         myParnterNetworkTeamCount[_newPartner] += 1;
669     }
670 }
```

Risk Level:

Likelihood – 4
Impact - 3

Recommendation:

To minimize the risk, consider implementing a logic where the user can request another user to be his partner; the partnership only occurs if the invitation is accepted.

Status - Mitigated

The Defibulls team has mitigated the risk by requiring the referral address to have a non-zero balance of BTC Bulls.

A.6 Race Condition [MEDIUM]

Description:

The mintingCost variable can be modified by the owner. If the user calls mint function and then the owner change the mintingCost using setMintingCost function, the mint function

call may get executed using the new mintingCost value if the owner's call gets executed first.

Code:

Listing 6: TheRanchBTCBullsCommunity.sol

```
241 function mint(uint256 _tokenQuantity, bool _enterRaffle) public payable
        ↪ {
242     if (paused) { revert Contract_CurrentlyPaused_CheckSocials();}
243     if (!publicSaleLive) { revert Minting_PublicSaleNotLive();}
244     if (_tokenQuantity == 0  _tokenQuantity > 10) { revert
            ↪ Minting_IsZeroOrBiggerThanTen();}
245     if (_tokenSupply.current() + _tokenQuantity > maxSupply) {revert
            ↪ Minting_ExceedsTotalBulls();}
246     if (addressMintCount[msg.sender] + _tokenQuantity >
            ↪ nftPerAddressLimit) { revert
            ↪ Minting_ExceedsMintingLimitPerAddress();}


249     IERC20Upgradeable usdcToken = IERC20Upgradeable(usdcTokenContract);
250     uint256 minting_cost_per_bull = mintingCost * 10 **
            ↪ usdcTokenDecimals;
251     uint256 totalTransactionCost = minting_cost_per_bull *
            ↪ _tokenQuantity;
252     usdcToken.safeTransferFrom(msg.sender, address(this), (
            ↪ totalTransactionCost));
```

Risk Level:

Likelihood – 2
Impact - 4

Recommendation:

Consider adding the price as an argument to the mint function, then verifying it to be the same as the one stored in the smart contract.

Status - Fixed

The DefiBulls team has fixed the issue by setting the mintingCost variable as constant.

A.7 Centralization Risk [MEDIUM]

Description:

The owner can blacklist any address using the blacklistMalicious function, this action can prevent the user from withdrawing his WBTC and USDC balance, and it represents a significant centralization risk where the owner have too much power on the users.

Code:

Listing 7: TheRanchBTCBullsCommunity.sol

```
915 function blacklistMalicious(address _address, bool value) external
        ↪ onlyOwner {
916     isBlacklisted[_address] = value;
917 }
```

Risk Level:

Likelihood – 2
Impact - 4

Recommendation:

Consider using a multisig wallet to include multiple parties in the blacklist functionality to avoid centralization risks. It is also recommended to notify the community of this behavior.

Status - Acknowledged

The DefiBulls team has acknowledged the issue, stating that the owner of the contract will be a multisig wallet.

A.8 Avoid using .transfer() to transfer Ether [LOW]

Description:

Although transfer() and send() are recommended as a security best-practice to prevent reentrancy attacks because they only forward 2300 gas, the gas repricing of opcodes may break deployed contracts.

Code:

Listing 8: TheRanchBTCBullsCommunity.sol

```
676 function withdraw() external onlyOwner {
677     payable(msg.sender).transfer(address(this).balance);
678 }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

Consider using .call{value: ...}("") instead, without hardcoded gas limits along with checks-effects-interactions pattern or reentrancy guards for reentrancy protection.

Status - Fixed

The DefiBulls team has fixed the issue by using .call{value: ...}("") instead of .transfer(),and changing the withdraw() function name to withdrawNativeToken().

A.9 For Loop Over Dynamic Array [LOW]

Description:

When smart contracts are deployed or their associated functions are invoked, the execution of these operations always consumes a certain quantity of gas, according to the amount of

computation required to accomplish them. Modifying an unknown-size array that grows in size over time can result in a Denial of Service. Simply by having an excessively huge array, users can exceed the gas limit, therefore preventing the transaction from ever succeeding.

Code:

Listing 9: TheRanchBTCBullsCommunity.sol

```
374 function updateMaintenanceStanding() external ADMIN_OR_DEFENDER {
375     for( uint i; i < rewardedAddresses.length; i++) {
376         address _wallet = rewardedAddresses[i];
377         if (btcBullOwners[_wallet].WBTC_Balance > 0){
378             if (btcBullOwners[_wallet].lastRewardDate !=
                ↪ currentRewardingDate) {

380                 // take action and add one to maintenanceFeesStanding
381                 btcBullOwners[_wallet].maintenanceFeesStanding += 1;

383                 if (btcBullOwners[_wallet].maintenanceFeesStanding == 4){
384                     upForLiquidation.push(_wallet);
385                 }
386             }
387         }
388     }
389 }
```

Listing 10: TheRanchBTCBullsCommunity.sol

```
543 for( uint i; i < upForLiquidation.length; i++) {
544     address _culprit = upForLiquidation[i];
545     uint256 _amount = btcBullOwners[_culprit].WBTC_Balance;
546     btcBullOwners[_culprit].WBTC_Balance = 0;
547     totalAmountLiquidated += _amount;

549     // reset fees and months behind.
550     btcBullOwners[_culprit].maintenanceFeeBalance = 0;
551     btcBullOwners[_culprit].maintenanceFeesStanding = 0;

553     // emit event
```

```
554     emit liquidationEvent(_culprit, _amount) ;

556 }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

Avoid actions that involve looping across the entire data structure. If you really must loop over an array of unknown size, arrange for it to consume many blocs and thus multiple transactions.

Status - Acknowledged

The DefiBulls team has acknowledged the risk.

A.10 Owner Can Renounce Ownership [LOW]

Description:

Typically, the account that deploys the contract is also its owner. Consequently, the owner is able to engage in certain privileged activities in his own name. In smart contracts, the renounceOwnership function is used to renounce ownership, which means that if the contract's ownership has never been transferred, it will never have an Owner, rendering some owner-exclusive functionality unavailable.

Code:

Listing 11: TheRanchBTCBullsCommunity.sol

```
40 contract TheRanchBTCBullsCommunity is
41     Initializable,
42     ERC721EnumerableUpgradeable,
```

```
43      OwnableUpgradeable,
44      ReentrancyGuardUpgradeable,
45      IERC2981Upgradeable
46      {
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

We recommend that you prevent the owner from calling renounceOwnership without first transferring ownership to a different address. Additionally, if you decide to use a multi-signature wallet, then the execution of the renounceOwnership will require for at least two or more users to be confirmed. Alternatively, you can disable Renounce Ownership functionality by overriding it.

Status - Fixed

The DefiBulls team has fixed the issue by overriding the renounceOwnership function to disable it.

A.11 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.0. Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps to ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 12: TheRanchBTCBullsCommunity.sol

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
```

Risk Level:

Likelihood – 0

Impact - 0

Recommendation:

Consider locking the pragma version. It is advised that floating pragma not be used in production.Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status - Fixed

The DefiBulls team has fixed the issue by fixing the pragma version to 0.8.7.

4 Best Practices

BP.1 safeTransfer Can Be Used Instead Of approve And transferFrom

Description:

To withdraw a balance from the contract to an external address a safeTransfer can be used instead of approving an amount then calling the safeTransferFrom function.

Code:

Listing 13: TheRanchBTCBullsCommunity.sol

```
684 function withdrawBtcMinersSafeBalance() external ADMIN_OR_DEFENDER {
685     IERC20Upgradeable tokenContract = IERC20Upgradeable(
        ↪ usdcTokenContract);
686     uint256 amtToTransfer = btcMinersSafeBalance;
687     tokenContract.approve(address(this), amtToTransfer);
688     tokenContract.safeTransferFrom(address(this), btcMinersSafe,
        ↪ amtToTransfer);
689     btcMinersSafeBalance -= amtToTransfer;
690
691
692 }
```

Code:

Listing 14: TheRanchBTCBullsCommunity.sol

```
694 function withdrawHostingSafeBalance() external ADMIN_OR_DEFENDER {
695     IERC20Upgradeable tokenContract = IERC20Upgradeable(
        ↪ usdcTokenContract);
696     uint256 amtToTransfer = hostingSafeBalance;
697     tokenContract.approve(address(this), amtToTransfer);
698     tokenContract.safeTransferFrom(address(this), hostingSafe,
        ↪ amtToTransfer);
```

```
699         hostingSafeBalance -= amtToTransfer;
700     }
```

BP.2 State Variables That Could Be Declared Immutable

Description:

The constant state variable herewith should be declared immutable to save gas.

Code:

Listing 15: TheRanchBTCBullsCommunity.sol

```
69  uint public constant maxSupply = 10000;
```

BP.3 Public Function Can Be Called External

Description:

Functions with a public scope that are not called inside the contract should be declared external to reduce the gas fees. (,788,795,802,820,824,833,853,897,902,906,911,935)

Code:

Listing 16: TheRanchBTCBullsCommunity.sol

```
241 function mint(uint256 _tokenQuantity, bool _enterRaffle) public payable
        ↪ {
```

Listing 17: TheRanchBTCBullsCommunity.sol

```
340 function setPayPerNftForTheMonthAndCurrentRewardingDate(uint256
        ↪ _totalAmountToDeposit, uint _dateOfRewarding) public onlyOwner {
```

Listing 18: TheRanchBTCBullsCommunity.sol

```
529 function getLiquidatedArrayLength() public view ADMIN_OR_DEFENDER
        ↪ returns (uint) {
```

Listing 19: TheRanchBTCBullsCommunity.sol

```
655 function setPartnerAddress(address _newPartner) public {
```

Listing 20: TheRanchBTCBullsCommunity.sol

```
749 function getRewardAddressesLength() public view returns (uint){
```

Listing 21: TheRanchBTCBullsCommunity.sol

```
753 function getMaintenanceFeeBalanceForAddress() public view returns (
        ↪ uint256){
```

Listing 22: TheRanchBTCBullsCommunity.sol

```
757 function getMaintenanceFeeStandingForAddress() public view returns (uint
        ↪ ){
```

Listing 23: TheRanchBTCBullsCommunity.sol

```
762 function getWbtcBalanceForAddress() public view returns (uint256){
```

Listing 24: TheRanchBTCBullsCommunity.sol

```
767 function getUsdcBalanceForAddress() public view returns (uint256) {
```

Listing 25: TheRanchBTCBullsCommunity.sol

```
774 function getPartnerNetworkTeamCount() public view returns (uint) {
```

Listing 26: TheRanchBTCBullsCommunity.sol

```
781 function getAreTheyOnMyPartnerNetworkTeam(address _adressToCheck) public
        ↪  view returns (bool) {
```

Listing 27: TheRanchBTCBullsCommunity.sol

```
788 function getRafflePlayer(uint256 index) public view onlyOwner returns (
        ↪ address) {
```

Listing 28: TheRanchBTCBullsCommunity.sol

```
795 function getHaveTheyMintedBefore(address _adressToCheck) public view
        ↪ returns (bool) {
```

Listing 29: TheRanchBTCBullsCommunity.sol

```
802 function getMintCountForAddress(address _address) public view returns (
        ↪ uint) {
```

Listing 30: TheRanchBTCBullsCommunity.sol

```
820 function getNumberOfRafflePlayers() public view returns (uint256) {
```

Listing 31: TheRanchBTCBullsCommunity.sol

```
824 function getBlacklistedStatus(address _address) public view returns (
        ↪ bool) {
```

Listing 32: TheRanchBTCBullsCommunity.sol

```
833 function tokenURI(uint256 tokenId) public view virtual override returns
        ↪ (string memory) {
```

Listing 33: TheRanchBTCBullsCommunity.sol

```
853 function setBaseURI(string memory _newBaseURI) public onlyOwner{
```

Listing 34: TheRanchBTCBullsCommunity.sol

```
897 function setUsdcTokenAddress(address _address) public onlyOwner {
```

Listing 35: TheRanchBTCBullsCommunity.sol

```
902 function setUsdcTokenDecimals(uint _decimals) public onlyOwner {
```

Listing 36: TheRanchBTCBullsCommunity.sol

```
906 function setWbtcTokenAddress(address _address) public onlyOwner {
```

Listing 37: TheRanchBTCBullsCommunity.sol

```
911 function setWbtcTokenDecimals(uint _decimals) public onlyOwner {
```

Listing 38: TheRanchBTCBullsCommunity.sol

```
935 function setStockYardInfo(uint _stockyardNumber, uint _startingIndex,
        ↪ uint _endingIndex) public onlyOwner {
```

5 Tests

Results:

```
tests/unit/test_TRBCProxy_ExceedingMaxMintLimit.py . [ 4%]
tests/unit/test_TRBCProxy_HaveTheyMintedBefore.py . [ 9%]
tests/unit/test_TRBCProxy_PartnerNetworkTeam.py . [ 14%]
tests/unit/test_TRBCProxy_admin_control.py . [ 19%]
tests/unit/test_TRBCProxy_balances_withdraw.py . [ 23%]
tests/unit/test_TRBCProxy_blacklist_usdc.py . [ 28%]
tests/unit/test_TRBCProxy_blacklist_wbtc.py . [ 33%]
tests/unit/test_TRBCProxy_double_raffle_entry.py . [ 38%]
tests/unit/test_TRBCProxy_feedingContract_works_correctly.py . [ 42%]
tests/unit/
   ↪ test_TRBCProxy_feedingContract_works_correctly_with_partner_set.
   ↪ py . [ 47%]
tests/unit/test_TRBCProxy_gas_reward_30_1mintEach_4months.py . [ 52%]
tests/unit/test_TRBCProxy_gas_reward_30_1mintEach_with_transfers.py . [
   ↪ 57%]
tests/unit/test_TRBCProxy_maint_fee_payment.py . [ 61%]
tests/unit/test_TRBCProxy_maintenance_fees.py . [ 66%]
tests/unit/test_TRBCProxy_minting_Raffle.py . [ 71%]
tests/unit/test_TRBCProxy_pause_and_mintingCost.py . [ 76%]
tests/unit/test_TRBCProxy_safe_addresses.py . [ 80%]
tests/unit/test_TRBCProxy_storage.py . [ 85%]
tests/unit/test_TRBCProxy_updateMaintenanceStanding.py . [ 90%]
tests/unit/test_usdc.py . [ 95%]
tests/unit/test_wbtc.py . [100%]

=============================================================================
   ↪  warnings summary
   ↪ =============================================================================
   ↪
tests/unit/test_TRBCProxy_ExceedingMaxMintLimit.py: 92 warnings
tests/unit/test_TRBCProxy_HaveTheyMintedBefore.py: 138 warnings
tests/unit/test_TRBCProxy_PartnerNetworkTeam.py: 194 warnings
tests/unit/test_TRBCProxy_admin_control.py: 37 warnings
tests/unit/test_TRBCProxy_balances_withdraw.py: 139 warnings
```

```
tests/unit/test_TRBCProxy_blacklist_usdc.py: 149 warnings
tests/unit/test_TRBCProxy_blacklist_wbtc.py: 186 warnings
tests/unit/test_TRBCProxy_double_raffle_entry.py: 124 warnings
tests/unit/test_TRBCProxy_feedingContract_works_correctly.py: 183
   ↪ warnings
tests/unit/
   ↪ test_TRBCProxy_feedingContract_works_correctly_with_partner_set.
   ↪ py: 174 warnings
tests/unit/test_TRBCProxy_gas_reward_30_1mintEach_4months.py: 835
   ↪ warnings
tests/unit/test_TRBCProxy_gas_reward_30_1mintEach_with_transfers.py: 444
   ↪  warnings
tests/unit/test_TRBCProxy_maint_fee_payment.py: 240 warnings
tests/unit/test_TRBCProxy_maintenance_fees.py: 318 warnings
tests/unit/test_TRBCProxy_minting_Raffle.py: 322 warnings
tests/unit/test_TRBCProxy_pause_and_mintingCost.py: 65 warnings
tests/unit/test_TRBCProxy_safe_addresses.py: 61 warnings
tests/unit/test_TRBCProxy_storage.py: 557 warnings
tests/unit/test_TRBCProxy_updateMaintenanceStanding.py: 305 warnings
tests/unit/test_usdc.py: 7 warnings
tests/unit/test_wbtc.py: 7 warnings
  DeprecationWarning: abi.encode_abi() and abi.encode_abi_packed() are
     ↪ deprecated and will be removed in version 4.0.0 in favor of abi.
     ↪ encode() and abi.encode_packed(), respectively
    warnings.warn(
tests/unit/test_TRBCProxy_ExceedingMaxMintLimit.py: 77 warnings
tests/unit/test_TRBCProxy_HaveTheyMintedBefore.py: 82 warnings
tests/unit/test_TRBCProxy_PartnerNetworkTeam.py: 137 warnings
tests/unit/test_TRBCProxy_admin_control.py: 13 warnings
tests/unit/test_TRBCProxy_balances_withdraw.py: 107 warnings
tests/unit/test_TRBCProxy_blacklist_usdc.py: 77 warnings
tests/unit/test_TRBCProxy_blacklist_wbtc.py: 111 warnings
tests/unit/test_TRBCProxy_double_raffle_entry.py: 74 warnings
tests/unit/test_TRBCProxy_feedingContract_works_correctly.py: 131
   ↪ warnings
tests/unit/
   ↪ test_TRBCProxy_feedingContract_works_correctly_with_partner_set.
   ↪ py: 120 warnings
```

```
tests/unit/test_TRBCProxy_gas_reward_30_1mintEach_4months.py: 729
   ↪ warnings
tests/unit/test_TRBCProxy_gas_reward_30_1mintEach_with_transfers.py: 322
   ↪  warnings
tests/unit/test_TRBCProxy_maint_fee_payment.py: 181 warnings
tests/unit/test_TRBCProxy_maintenance_fees.py: 250 warnings
tests/unit/test_TRBCProxy_minting_Raffle.py: 209 warnings
tests/unit/test_TRBCProxy_pause_and_mintingCost.py: 47 warnings
tests/unit/test_TRBCProxy_safe_addresses.py: 49 warnings
tests/unit/test_TRBCProxy_storage.py: 420 warnings
tests/unit/test_TRBCProxy_updateMaintenanceStanding.py: 228 warnings
tests/unit/test_usdc.py: 5 warnings
tests/unit/test_wbtc.py: 5 warnings
  DeprecationWarning: abi.decode_abi() is deprecated and will be removed
     ↪  in version 4.0.0 in favor of abi.decode()
    warnings.warn(
tests/unit/test_TRBCProxy_ExceedingMaxMintLimit.py: 36 warnings
tests/unit/test_TRBCProxy_balances_withdraw.py: 12 warnings
tests/unit/test_TRBCProxy_blacklist_usdc.py: 4 warnings
tests/unit/test_TRBCProxy_blacklist_wbtc.py: 17 warnings
tests/unit/test_TRBCProxy_gas_reward_30_1mintEach_4months.py: 103
   ↪ warnings
tests/unit/test_TRBCProxy_gas_reward_30_1mintEach_with_transfers.py: 8
   ↪ warnings
tests/unit/test_TRBCProxy_maint_fee_payment.py: 23 warnings
tests/unit/test_TRBCProxy_maintenance_fees.py: 45 warnings
tests/unit/test_TRBCProxy_minting_Raffle.py: 2 warnings
tests/unit/test_TRBCProxy_storage.py: 8 warnings
tests/unit/test_TRBCProxy_updateMaintenanceStanding.py: 24 warnings
  DeprecationWarning: abi.decode_single() is deprecated and will be
     ↪ removed in version 4.0.0 in favor of abi.decode()
    warnings.warn(
-- Docs: https://docs.pytest.org/en/stable/warnings.html
============================================================ 21 passed,
   ↪ 8233 warnings in 719.62s (0:11:59)
   ↪ ============================================================
Terminating local RPC client...
```

6 Static Analysis (Slither)

Description:

ShellBoxes expanded the coverage of the specific contract areas using automated testing methodologies. Slither, a Solidity static analysis framework, was one of the tools used. Slither was run on all-scoped contracts in both text and binary formats. This tool can be used to test mathematical relationships between Solidity instances statically and variables that allow for the detection of errors or inconsistent usage of the contracts' APIs throughout the entire codebase.

Results:

```
Compilation warnings/errors on contracts/TheRanchBTCBullsCommunity.sol:
Warning: Contract code size exceeds 24576 bytes (a limit introduced in
    ↪ Spurious Dragon). This contract may not be deployable on mainnet.
    ↪  Consider enabling the optimizer (with a low "runs" value!),
    ↪ turning off revert strings, or using libraries.
  --> contracts/TheRanchBTCBullsCommunity.sol:40:1:
   |
40 | contract TheRanchBTCBullsCommunity is
   | ^ (Relevant source part starts here and spans across multiple lines
      ↪ ).

TheRanchBTCBullsCommunity.mint(uint256,bool) (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#241-301) performs a multiplication
    ↪ on the result of a division:
        -referralFundAmt = totalTransactionCost * 2 / 100 (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#278)
        -splitReferralAmt = referralFundAmt * 50 / 100 (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#293)
TheRanchBTCBullsCommunity.rewardBulls(uint256) (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#406-477) performs a multiplication
    ↪ on the result of a division:
        -referralAmt = totalPayoutForTheBullOwner * 1 / 100 (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#445)
```

```
        -_bullOwner.WBTC_Balance += (totalPayoutForTheBullOwner - (
            ↪ referralAmt * 2)) (contracts/TheRanchBTCBullsCommunity.sol
            ↪ #456)
TheRanchBTCBullsCommunity.updateUsdcBonusFromAnotherContract(address[],
    ↪ uint256) (contracts/TheRanchBTCBullsCommunity.sol#490-525)
    ↪ performs a multiplication on the result of a division:
        -deductionAmt = _amountToAdd * 5 / 100 (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#517)
        -hostingSafeBalance += (deductionAmt * 4) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#521)
TheRanchBTCBullsCommunity.updateUsdcBonusFromAnotherContract(address[],
    ↪ uint256) (contracts/TheRanchBTCBullsCommunity.sol#490-525)
    ↪ performs a multiplication on the result of a division:
        -referralAmt = _amountToAdd * 5 / 100 (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#502)
        -btcBullOwners[_ownerOfNFT].USDC_Balance += (_amountToAdd - (
            ↪ referralAmt * 2)) (contracts/TheRanchBTCBullsCommunity.sol
            ↪ #512)
TheRanchBTCBullsCommunity.updateUsdcBonusFromAnotherContract(address[],
    ↪ uint256) (contracts/TheRanchBTCBullsCommunity.sol#490-525)
    ↪ performs a multiplication on the result of a division:
        -deductionAmt = _amountToAdd * 5 / 100 (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#517)
        -btcBullOwners[_ownerOfNFT].USDC_Balance += (_amountToAdd - (
            ↪ deductionAmt * 6)) (contracts/TheRanchBTCBullsCommunity.
            ↪ sol#522)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #divide-before-multiply

Reentrancy in TheRanchBTCBullsCommunity.mint(uint256,bool) (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#241-301):
        External calls:
        - usdcToken.safeTransferFrom(msg.sender,address(this),(
            ↪ totalTransactionCost)) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#252)
        State variables written after the call(s):
        - addressMintCount[msg.sender] += 1 (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#256)
```

```
Reentrancy in TheRanchBTCBullsCommunity.payMaintanenceFees() (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#572-609):
        External calls:
        - usdcToken.safeTransferFrom(msg.sender,address(this),(_feesDue))
            ↪  (contracts/TheRanchBTCBullsCommunity.sol#590)
        - usdcToken_scope_0.safeTransferFrom(msg.sender,address(this),(
            ↪ amt_needed)) (contracts/TheRanchBTCBullsCommunity.sol#597)
        State variables written after the call(s):
        - btcBullOwners[msg.sender].maintenanceFeeBalance = 0 (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#606)
        - btcBullOwners[msg.sender].maintenanceFeesStanding = 0 (
            ↪ contracts/TheRanchBTCBullsCommunity.sol#607)
Reentrancy in TheRanchBTCBullsCommunity.
    ↪ setPayPerNftForTheMonthAndCurrentRewardingDate(uint256,uint256) (
    ↪ contracts/TheRanchBTCBullsCommunity.sol#340-367):
        External calls:
        - tokenContract.safeTransferFrom(msg.sender,address(this),
            ↪ _totalAmountToDeposit) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#346)
        State variables written after the call(s):
        - lastDeposit = _totalAmountToDeposit (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#349)
Reentrancy in TheRanchBTCBullsCommunity.withdrawBtcMinersSafeBalance() (
    ↪ contracts/TheRanchBTCBullsCommunity.sol#684-692):
        External calls:
        - tokenContract.approve(address(this),amtToTransfer) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#687)
        - tokenContract.safeTransferFrom(address(this),btcMinersSafe,
            ↪ amtToTransfer) (contracts/TheRanchBTCBullsCommunity.sol
            ↪ #688)
        State variables written after the call(s):
        - btcMinersSafeBalance -= amtToTransfer (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#689)
Reentrancy in TheRanchBTCBullsCommunity.withdrawHostingSafeBalance() (
    ↪ contracts/TheRanchBTCBullsCommunity.sol#694-700):
        External calls:
        - tokenContract.approve(address(this),amtToTransfer) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#697)
```

```
        - tokenContract.safeTransferFrom(address(this),hostingSafe,
            ↪ amtToTransfer) (contracts/TheRanchBTCBullsCommunity.sol
            ↪ #698)
        State variables written after the call(s):
        - hostingSafeBalance -= amtToTransfer (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#699)
Reentrancy in TheRanchBTCBullsCommunity.withdrawUsdcBalance() (contracts
    ↪ /TheRanchBTCBullsCommunity.sol#722-742):
        External calls:
        - IERC20Upgradeable(usdcTokenContract).safeTransfer(msg.sender,(
            ↪ myBalance)) (contracts/TheRanchBTCBullsCommunity.sol#731)
        State variables written after the call(s):
        - btcBullOwners[msg.sender].USDC_Balance = 0 (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#734)
Reentrancy in TheRanchBTCBullsCommunity.withdrawWbtcBalance() (contracts
    ↪ /TheRanchBTCBullsCommunity.sol#703-720):
        External calls:
        - IERC20Upgradeable(wbtcTokenContract).safeTransfer(msg.sender,
            ↪ myBalance) (contracts/TheRanchBTCBullsCommunity.sol#714)
        State variables written after the call(s):
        - btcBullOwners[msg.sender].WBTC_Balance = 0 (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#717)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-1

TheRanchBTCBullsCommunity.updateUsdcBonusFromAnotherContract(address[],
    ↪ uint256).i (contracts/TheRanchBTCBullsCommunity.sol#493) is a
    ↪ local variable never initialized
TheRanchBTCBullsCommunity.updateMaintenanceStanding().i (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#375) is a local variable never
    ↪ initialized
TheRanchBTCBullsCommunity.walletOfOwner(address).i (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#813) is a local variable never
    ↪ initialized
TheRanchBTCBullsCommunity.liquidateOutstandingAccounts().i (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#543) is a local variable never
    ↪ initialized
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #uninitialized-local-variables

TheRanchBTCBullsCommunity.liquidateOutstandingAccounts() (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#537-563) ignores return value by
    ↪ tokenContract.approve(address(this),totalAmountLiquidated) (
    ↪ contracts/TheRanchBTCBullsCommunity.sol#561)
TheRanchBTCBullsCommunity.withdrawBtcMinersSafeBalance() (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#684-692) ignores return value by
    ↪ tokenContract.approve(address(this),amtToTransfer) (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#687)
TheRanchBTCBullsCommunity.withdrawHostingSafeBalance() (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#694-700) ignores return value by
    ↪ tokenContract.approve(address(this),amtToTransfer) (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#697)
ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes)
    ↪ (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/
    ↪ ERC721Upgradeable.sol#399-421) ignores return value by
    ↪ IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from
    ↪ ,tokenId,data) (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ token/ERC721/ERC721Upgradeable.sol#406-417)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-return

TheRanchBTCBullsCommunity.walletOfOwner(address)._owner (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#810) shadows:
        - OwnableUpgradeable._owner (node_modules/@openzeppelin/contracts
          ↪ -upgradeable/access/OwnableUpgradeable.sol#22) (state
          ↪ variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #local-variable-shadowing

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes)
    ↪ (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/
    ↪ ERC721Upgradeable.sol#399-421) has external calls inside a loop:
    ↪ IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from
    ↪ ,tokenId,data) (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ token/ERC721/ERC721Upgradeable.sol#406-417)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ /#calls-inside-a-loop

Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,
    ↪ uint256,bytes).retval (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/token/ERC721/ERC721Upgradeable.sol#406)' in
    ↪ ERC721Upgradeable._checkOnERC721Received(address,address,uint256,
    ↪ bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/
    ↪ ERC721/ERC721Upgradeable.sol#399-421) potentially used before
    ↪ declaration: retval == IERC721ReceiverUpgradeable.
    ↪ onERC721Received.selector (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/token/ERC721/ERC721Upgradeable.sol#407)
Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,
    ↪ uint256,bytes).reason (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/token/ERC721/ERC721Upgradeable.sol#408)' in
    ↪ ERC721Upgradeable._checkOnERC721Received(address,address,uint256,
    ↪ bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/
    ↪ ERC721/ERC721Upgradeable.sol#399-421) potentially used before
    ↪ declaration: reason.length == 0 (node_modules/@openzeppelin/
    ↪ contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#409)
Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,
    ↪ uint256,bytes).reason (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/token/ERC721/ERC721Upgradeable.sol#408)' in
    ↪ ERC721Upgradeable._checkOnERC721Received(address,address,uint256,
    ↪ bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/
    ↪ ERC721/ERC721Upgradeable.sol#399-421) potentially used before
    ↪ declaration: revert(uint256,uint256)(32 + reason,mload(uint256)(
    ↪ reason)) (node_modules/@openzeppelin/contracts-upgradeable/token/
    ↪ ERC721/ERC721Upgradeable.sol#414)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #pre-declaration-usage-of-local-variables

Reentrancy in TheRanchBTCBullsCommunity.mint(uint256,bool) (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#241-301):
        External calls:
        - usdcToken.safeTransferFrom(msg.sender,address(this),(
            ↪ totalTransactionCost)) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#252)
```

```
        State variables written after the call(s):
        - USDCRewardsBalance += (referralFundAmt + raffleFundAmt) (
            ↪ contracts/TheRanchBTCBullsCommunity.sol#281)
        - btcBullOwners[referrer].USDC_Balance += referralFundAmt (
            ↪ contracts/TheRanchBTCBullsCommunity.sol#289)
        - btcBullOwners[coreTeam_1].USDC_Balance += splitReferralAmt (
            ↪ contracts/TheRanchBTCBullsCommunity.sol#296)
        - btcBullOwners[coreTeam_2].USDC_Balance += splitReferralAmt (
            ↪ contracts/TheRanchBTCBullsCommunity.sol#297)
        - btcMinersSafeBalance += btcMinersSafeAmt (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#282)
        - dailyRaffleBalance += raffleFundAmt (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#272)
        - dailyRafflePlayers.push(address(msg.sender)) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#269)
        - hostingSafeBalance += hostingSafeAmt (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#283)
        - userInDailyRaffle[msg.sender] = true (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#270)
        - userMintCount[msg.sender] += _tokenQuantity (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#261)
    Reentrancy in TheRanchBTCBullsCommunity.payMaintanenceFees() (contracts/
        ↪ TheRanchBTCBullsCommunity.sol#572-609):
        External calls:
        - usdcToken.safeTransferFrom(msg.sender,address(this),(_feesDue))
            ↪  (contracts/TheRanchBTCBullsCommunity.sol#590)
        State variables written after the call(s):
        - hostingSafeBalance += _feesDue (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#591)
    Reentrancy in TheRanchBTCBullsCommunity.payMaintanenceFees() (contracts/
        ↪ TheRanchBTCBullsCommunity.sol#572-609):
        External calls:
        - usdcToken_scope_0.safeTransferFrom(msg.sender,address(this),(
            ↪ amt_needed)) (contracts/TheRanchBTCBullsCommunity.sol#597)
        State variables written after the call(s):
        - hostingSafeBalance += (amt_needed + _balance) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#599)
```

```
Reentrancy in TheRanchBTCBullsCommunity.
    ↪ setPayPerNftForTheMonthAndCurrentRewardingDate(uint256,uint256) (
    ↪ contracts/TheRanchBTCBullsCommunity.sol#340-367):
        External calls:
        - tokenContract.safeTransferFrom(msg.sender,address(this),
            ↪ _totalAmountToDeposit) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#346)
        State variables written after the call(s):
        - btcBullOwners[coreTeam_1].WBTC_Balance += coreTeam_1_amt (
            ↪ contracts/TheRanchBTCBullsCommunity.sol#360)
        - btcBullOwners[coreTeam_2].WBTC_Balance += coreTeam_2_amt (
            ↪ contracts/TheRanchBTCBullsCommunity.sol#361)
        - currentRewardingDate = _dateOfRewarding (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#348)
        - payPerNftForTheMonth = payout_per_nft (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#358)
Reentrancy in TheRanchBTCBullsCommunity.withdrawUsdcBalance() (contracts
    ↪ /TheRanchBTCBullsCommunity.sol#722-742):
        External calls:
        - IERC20Upgradeable(usdcTokenContract).safeTransfer(msg.sender,(
            ↪ myBalance)) (contracts/TheRanchBTCBullsCommunity.sol#731)
        State variables written after the call(s):
        - USDCRewardsBalance -= myBalance (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#737)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-2

Reentrancy in TheRanchBTCBullsCommunity.mint(uint256,bool) (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#241-301):
        External calls:
        - usdcToken.safeTransferFrom(msg.sender,address(this),(
            ↪ totalTransactionCost)) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#252)
        - _safeMint(msg.sender,_tokenSupply.current()) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#257)
                - IERC721ReceiverUpgradeable(to).onERC721Received(
                    ↪ _msgSender(),from,tokenId,data) (node_modules/
                    ↪ @openzeppelin/contracts-upgradeable/token/ERC721/
```

```
                ↪ ERC721Upgradeable.sol#406-417)
        Event emitted after the call(s):
        - Transfer(address(0),to,tokenId) (node_modules/@openzeppelin/
            ↪ contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol
            ↪ #293)
                - _safeMint(msg.sender,_tokenSupply.current()) (contracts/
                    ↪ TheRanchBTCBullsCommunity.sol#257)
Reentrancy in TheRanchBTCBullsCommunity.mint(uint256,bool) (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#241-301):
        External calls:
        - usdcToken.safeTransferFrom(msg.sender,address(this),(
            ↪ totalTransactionCost)) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#252)
        Event emitted after the call(s):
        - NewBullsEnteringRanch(msg.sender,_enterRaffle,_tokenQuantity,
            ↪ _tokenSupply.current()) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#300)
Reentrancy in TheRanchBTCBullsCommunity.payMaintanenceFees() (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#572-609):
        External calls:
        - usdcToken.safeTransferFrom(msg.sender,address(this),(_feesDue))
            ↪  (contracts/TheRanchBTCBullsCommunity.sol#590)
        - usdcToken_scope_0.safeTransferFrom(msg.sender,address(this),(
            ↪ amt_needed)) (contracts/TheRanchBTCBullsCommunity.sol#597)
        Event emitted after the call(s):
        - payMaintanenceFeesEvent(msg.sender,_balance,amt_needed) (
            ↪ contracts/TheRanchBTCBullsCommunity.sol#602)
Reentrancy in TheRanchBTCBullsCommunity.
    ↪ setPayPerNftForTheMonthAndCurrentRewardingDate(uint256,uint256) (
    ↪ contracts/TheRanchBTCBullsCommunity.sol#340-367):
        External calls:
        - tokenContract.safeTransferFrom(msg.sender,address(this),
            ↪ _totalAmountToDeposit) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#346)
        Event emitted after the call(s):
        - setPayPerNFTEvent(_totalAmountToDeposit,payout_per_nft,
            ↪ _dateOfRewarding) (contracts/TheRanchBTCBullsCommunity.sol
            ↪ #364)
```

```
Reentrancy in TheRanchBTCBullsCommunity.withdrawUsdcBalance() (contracts
    ↪ /TheRanchBTCBullsCommunity.sol#722-742):
        External calls:
        - IERC20Upgradeable(usdcTokenContract).safeTransfer(msg.sender,(
            ↪ myBalance)) (contracts/TheRanchBTCBullsCommunity.sol#731)
        Event emitted after the call(s):
        - withdrawUSDCBalanceForAddressEvent(msg.sender,myBalance) (
            ↪ contracts/TheRanchBTCBullsCommunity.sol#740)
Reentrancy in TheRanchBTCBullsCommunity.withdrawWbtcBalance() (contracts
    ↪ /TheRanchBTCBullsCommunity.sol#703-720):
        External calls:
        - IERC20Upgradeable(wbtcTokenContract).safeTransfer(msg.sender,
            ↪ myBalance) (contracts/TheRanchBTCBullsCommunity.sol#714)
        Event emitted after the call(s):
        - withdrawWbtcBalanceEvent(msg.sender,myBalance) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#719)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-3

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes)
    ↪ (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/
    ↪ ERC721Upgradeable.sol#399-421) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/
            ↪ token/ERC721/ERC721Upgradeable.sol#413-415)
AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol
    ↪ #174-194) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/
            ↪ utils/AddressUpgradeable.sol#186-189)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

TheRanchBTCBullsCommunity.mint(uint256,bool) (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#241-301) compares to a boolean
    ↪ constant:
        -_enterRaffle == true (contracts/TheRanchBTCBullsCommunity.sol
            ↪ #266)
```

```
TheRanchBTCBullsCommunity.mint(uint256,bool) (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#241-301) compares to a boolean
    ↪ constant:
        -getUserAlreadyInDailyRaffleStatus(msg.sender) == false (
            ↪ contracts/TheRanchBTCBullsCommunity.sol#268)
TheRanchBTCBullsCommunity.rewardBulls(uint256) (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#406-477) compares to a boolean
    ↪ constant:
        -readyToReward == false (contracts/TheRanchBTCBullsCommunity.sol
            ↪ #408)
TheRanchBTCBullsCommunity.updateUsdcBonusFromAnotherContract(address[],
    ↪ uint256) (contracts/TheRanchBTCBullsCommunity.sol#490-525)
    ↪ compares to a boolean constant:
        -require(bool,string)(isEcosystemRole[msg.sender] == true,must be
            ↪  approved to interact) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#491)
TheRanchBTCBullsCommunity.mintingRaffle(uint256) (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#621-639) compares to a boolean
    ↪ constant:
        -paused == false (contracts/TheRanchBTCBullsCommunity.sol#625)
TheRanchBTCBullsCommunity.mintingRaffle(uint256) (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#621-639) compares to a boolean
    ↪ constant:
        -require(bool,string)(isChainLinkVRFRole[msg.sender] == true,must
            ↪  be approved to interact) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#623)
TheRanchBTCBullsCommunity.ADMIN_OR_DEFENDER() (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#153-156) compares to a boolean
    ↪ constant:
        -require(bool,string)(msg.sender == owner() || isDefenderRole[msg
            ↪ .sender] == true,Caller is not an OWNER OR DEFENDER) (
            ↪ contracts/TheRanchBTCBullsCommunity.sol#154)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #boolean-equality

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.2']
        - ^0.8.0 (contracts/TheRanchBTCBullsCommunity.sol#2)
```

```
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access
  ↪ /OwnableUpgradeable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
  ↪ interfaces/IERC2981Upgradeable.sol#4)
- ^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/
  ↪ utils/Initializable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
  ↪ security/ReentrancyGuardUpgradeable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/
  ↪ ERC20/IERC20Upgradeable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/
  ↪ ERC20/extensions/draft-IERC20PermitUpgradeable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/
  ↪ ERC20/utils/SafeERC20Upgradeable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/
  ↪ ERC721/ERC721Upgradeable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/
  ↪ ERC721/IERC721ReceiverUpgradeable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/
  ↪ ERC721/IERC721Upgradeable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/
  ↪ ERC721/extensions/ERC721EnumerableUpgradeable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/
  ↪ ERC721/extensions/IERC721EnumerableUpgradeable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/
  ↪ ERC721/extensions/IERC721MetadataUpgradeable.sol#4)
- ^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/
  ↪ AddressUpgradeable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/
  ↪ ContextUpgradeable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/
  ↪ CountersUpgradeable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/
  ↪ StringsUpgradeable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/
  ↪ introspection/ERC165Upgradeable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/
  ↪ introspection/IERC165Upgradeable.sol#4)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #different-pragma-directives-are-used

ERC721EnumerableUpgradeable._removeTokenFromAllTokensEnumeration(uint256
    ↪ ) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/
    ↪ extensions/ERC721EnumerableUpgradeable.sol#150-168) has costly
    ↪ operations inside a loop:
        - delete _allTokensIndex[tokenId] (node_modules/@openzeppelin/
            ↪ contracts-upgradeable/token/ERC721/extensions/
            ↪ ERC721EnumerableUpgradeable.sol#166)
ERC721EnumerableUpgradeable._removeTokenFromAllTokensEnumeration(uint256
    ↪ ) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/
    ↪ extensions/ERC721EnumerableUpgradeable.sol#150-168) has costly
    ↪ operations inside a loop:
        - _allTokens.pop() (node_modules/@openzeppelin/contracts-
            ↪ upgradeable/token/ERC721/extensions/
            ↪ ERC721EnumerableUpgradeable.sol#167)
ERC721EnumerableUpgradeable._removeTokenFromOwnerEnumeration(address,
    ↪ uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/
    ↪ ERC721/extensions/ERC721EnumerableUpgradeable.sol#125-143) has
    ↪ costly operations inside a loop:
        - delete _ownedTokensIndex[tokenId] (node_modules/@openzeppelin/
            ↪ contracts-upgradeable/token/ERC721/extensions/
            ↪ ERC721EnumerableUpgradeable.sol#141)
TheRanchBTCBullsCommunity.updateUsdcBonusFromAnotherContract(address[],
    ↪ uint256) (contracts/TheRanchBTCBullsCommunity.sol#490-525) has
    ↪ costly operations inside a loop:
        - hostingSafeBalance += (deductionAmt * 4) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#521)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #costly-operations-inside-a-loop

AddressUpgradeable.functionCall(address,bytes) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol
    ↪ #85-87) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/utils/
    ↪ AddressUpgradeable.sol#114-120) is never used and should be
```

```
    ↪ removed
AddressUpgradeable.functionStaticCall(address,bytes) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol
    ↪ #147-149) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/utils/
    ↪ AddressUpgradeable.sol#157-166) is never used and should be
    ↪ removed
AddressUpgradeable.sendValue(address,uint256) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol
    ↪ #60-65) is never used and should be removed
ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/
    ↪ contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is
    ↪ never used and should be removed
ContextUpgradeable.__Context_init_unchained() (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol
    ↪ #21-22) is never used and should be removed
ContextUpgradeable._msgData() (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/utils/ContextUpgradeable.sol#27-29) is never used and
    ↪  should be removed
CountersUpgradeable.decrement(CountersUpgradeable.Counter) (node_modules
    ↪ /@openzeppelin/contracts-upgradeable/utils/CountersUpgradeable.
    ↪ sol#32-38) is never used and should be removed
CountersUpgradeable.reset(CountersUpgradeable.Counter) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/CountersUpgradeable.sol
    ↪ #40-42) is never used and should be removed
ERC165Upgradeable.__ERC165_init() (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/utils/introspection/ERC165Upgradeable.sol#24-25) is
    ↪ never used and should be removed
ERC165Upgradeable.__ERC165_init_unchained() (node_modules/@openzeppelin/
    ↪ contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol
    ↪ #27-28) is never used and should be removed
ERC721EnumerableUpgradeable.__ERC721Enumerable_init_unchained() (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/
    ↪ extensions/ERC721EnumerableUpgradeable.sol#19-20) is never used
    ↪ and should be removed
ERC721Upgradeable._baseURI() (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/token/ERC721/ERC721Upgradeable.sol#110-112) is never
```

```
    ↪ used and should be removed
ERC721Upgradeable._burn(uint256) (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/token/ERC721/ERC721Upgradeable.sol#308-322) is never
    ↪ used and should be removed
Initializable._disableInitializers() (node_modules/@openzeppelin/
    ↪ contracts-upgradeable/proxy/utils/Initializable.sol#131-137) is
    ↪ never used and should be removed
SafeERC20Upgradeable.safeApprove(IERC20Upgradeable,address,uint256) (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/
    ↪ utils/SafeERC20Upgradeable.sol#46-59) is never used and should be
    ↪  removed
SafeERC20Upgradeable.safeDecreaseAllowance(IERC20Upgradeable,address,
    ↪ uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/
    ↪ ERC20/utils/SafeERC20Upgradeable.sol#70-81) is never used and
    ↪ should be removed
SafeERC20Upgradeable.safeIncreaseAllowance(IERC20Upgradeable,address,
    ↪ uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/
    ↪ ERC20/utils/SafeERC20Upgradeable.sol#61-68) is never used and
    ↪ should be removed
SafeERC20Upgradeable.safePermit(IERC20PermitUpgradeable,address,address,
    ↪ uint256,uint256,uint8,bytes32,bytes32) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/token/ERC20/utils/
    ↪ SafeERC20Upgradeable.sol#83-97) is never used and should be
    ↪ removed
StringsUpgradeable.toHexString(address) (node_modules/@openzeppelin/
    ↪ contracts-upgradeable/utils/StringsUpgradeable.sol#72-74) is
    ↪ never used and should be removed
StringsUpgradeable.toHexString(uint256) (node_modules/@openzeppelin/
    ↪ contracts-upgradeable/utils/StringsUpgradeable.sol#41-52) is
    ↪ never used and should be removed
StringsUpgradeable.toHexString(uint256,uint256) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol
    ↪ #57-67) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Pragma version^0.8.0 (contracts/TheRanchBTCBullsCommunity.sol#2) allows
    ↪ old versions
```

```
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ access/OwnableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ interfaces/IERC2981Upgradeable.sol#4) allows old versions
Pragma version^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ security/ReentrancyGuardUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ token/ERC20/IERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ token/ERC20/extensions/draft-IERC20PermitUpgradeable.sol#4)
   ↪ allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ token/ERC20/utils/SafeERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ token/ERC721/ERC721Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ token/ERC721/IERC721ReceiverUpgradeable.sol#4) allows old
   ↪ versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ token/ERC721/IERC721Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ token/ERC721/extensions/ERC721EnumerableUpgradeable.sol#4) allows
   ↪  old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ token/ERC721/extensions/IERC721EnumerableUpgradeable.sol#4)
   ↪ allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ token/ERC721/extensions/IERC721MetadataUpgradeable.sol#4) allows
   ↪ old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ utils/CountersUpgradeable.sol#4) allows old versions
```

```
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ utils/StringsUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ utils/introspection/ERC165Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ utils/introspection/IERC165Upgradeable.sol#4) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/utils/
    ↪ AddressUpgradeable.sol#60-65):
        - (success) = recipient.call{value: amount}() (node_modules/
            ↪ @openzeppelin/contracts-upgradeable/utils/
            ↪ AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes
    ↪ ,uint256,string) (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/utils/AddressUpgradeable.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (
            ↪ node_modules/@openzeppelin/contracts-upgradeable/utils/
            ↪ AddressUpgradeable.sol#137)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,
    ↪ string) (node_modules/@openzeppelin/contracts-upgradeable/utils/
    ↪ AddressUpgradeable.sol#157-166):
        - (success,returndata) = target.staticcall(data) (node_modules/
            ↪ @openzeppelin/contracts-upgradeable/utils/
            ↪ AddressUpgradeable.sol#164)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls

Event TheRanchBTCBullsCommunitymintingRaffleEvent(uint256,address,
    ↪ uint256) (contracts/TheRanchBTCBullsCommunity.sol#168-173) is not
    ↪  in CapWords
Event TheRanchBTCBullsCommunitywithdrawUSDCBalanceForAddressEvent(
    ↪ address,uint256) (contracts/TheRanchBTCBullsCommunity.sol
    ↪ #175-178) is not in CapWords
Event TheRanchBTCBullsCommunitywithdrawWbtcBalanceEvent(address,uint256)
    ↪  (contracts/TheRanchBTCBullsCommunity.sol#180-183) is not in
```

```
    ↪ CapWords
Event TheRanchBTCBullsCommunityliquidationEvent(address,uint256) (
    ↪ contracts/TheRanchBTCBullsCommunity.sol#185-188) is not in
    ↪ CapWords
Event TheRanchBTCBullsCommunityrewardEvent(uint256,uint256,uint256,
    ↪ uint256) (contracts/TheRanchBTCBullsCommunity.sol#190-195) is not
    ↪  in CapWords
Event TheRanchBTCBullsCommunityresetRewardEvent(address,string) (
    ↪ contracts/TheRanchBTCBullsCommunity.sol#197-200) is not in
    ↪ CapWords
Event TheRanchBTCBullsCommunitysetPayPerNFTEvent(uint256,uint256,uint256
    ↪ ) (contracts/TheRanchBTCBullsCommunity.sol#202-206) is not in
    ↪ CapWords
Event TheRanchBTCBullsCommunitypayMaintanenceFeesEvent(address,uint256,
    ↪ uint256) (contracts/TheRanchBTCBullsCommunity.sol#209-213) is not
    ↪  in CapWords
Parameter TheRanchBTCBullsCommunity.mint(uint256,bool)._tokenQuantity (
    ↪ contracts/TheRanchBTCBullsCommunity.sol#241) is not in mixedCase
Parameter TheRanchBTCBullsCommunity.mint(uint256,bool)._enterRaffle (
    ↪ contracts/TheRanchBTCBullsCommunity.sol#241) is not in mixedCase
Parameter TheRanchBTCBullsCommunity.
    ↪ setPayPerNftForTheMonthAndCurrentRewardingDate(uint256,uint256).
    ↪ _totalAmountToDeposit (contracts/TheRanchBTCBullsCommunity.sol
    ↪ #340) is not in mixedCase
Parameter TheRanchBTCBullsCommunity.
    ↪ setPayPerNftForTheMonthAndCurrentRewardingDate(uint256,uint256).
    ↪ _dateOfRewarding (contracts/TheRanchBTCBullsCommunity.sol#340) is
    ↪  not in mixedCase
Parameter TheRanchBTCBullsCommunity.rewardBulls(uint256).
    ↪ _stockyardNumber (contracts/TheRanchBTCBullsCommunity.sol#406) is
    ↪  not in mixedCase
Parameter TheRanchBTCBullsCommunity.updateUsdcBonusFromAnotherContract(
    ↪ address[],uint256)._ownersOfTheNFTs (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#490) is not in mixedCase
Parameter TheRanchBTCBullsCommunity.updateUsdcBonusFromAnotherContract(
    ↪ address[],uint256)._amountToAdd (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#490) is not in mixedCase
```

```
Parameter TheRanchBTCBullsCommunity.mintingRaffle(uint256)._winningIndex
    ↪  (contracts/TheRanchBTCBullsCommunity.sol#621) is not in
    ↪ mixedCase
Parameter TheRanchBTCBullsCommunity.setPartnerAddress(address).
    ↪ _newPartner (contracts/TheRanchBTCBullsCommunity.sol#655) is not
    ↪ in mixedCase
Parameter TheRanchBTCBullsCommunity.withdrawToken(address,uint256).
    ↪ _tokenContract (contracts/TheRanchBTCBullsCommunity.sol#679) is
    ↪ not in mixedCase
Parameter TheRanchBTCBullsCommunity.withdrawToken(address,uint256).
    ↪ _amount (contracts/TheRanchBTCBullsCommunity.sol#679) is not in
    ↪ mixedCase
Parameter TheRanchBTCBullsCommunity.getAreTheyOnMyPartnerNetworkTeam(
    ↪ address)._adressToCheck (contracts/TheRanchBTCBullsCommunity.sol
    ↪ #781) is not in mixedCase
Parameter TheRanchBTCBullsCommunity.getHaveTheyMintedBefore(address).
    ↪ _adressToCheck (contracts/TheRanchBTCBullsCommunity.sol#795) is
    ↪ not in mixedCase
Parameter TheRanchBTCBullsCommunity.getMintCountForAddress(address).
    ↪ _address (contracts/TheRanchBTCBullsCommunity.sol#802) is not in
    ↪ mixedCase
Parameter TheRanchBTCBullsCommunity.getUserAlreadyInDailyRaffleStatus(
    ↪ address)._address (contracts/TheRanchBTCBullsCommunity.sol#806)
    ↪ is not in mixedCase
Parameter TheRanchBTCBullsCommunity.walletOfOwner(address)._owner (
    ↪ contracts/TheRanchBTCBullsCommunity.sol#810) is not in mixedCase
Parameter TheRanchBTCBullsCommunity.getBlacklistedStatus(address).
    ↪ _address (contracts/TheRanchBTCBullsCommunity.sol#824) is not in
    ↪ mixedCase
Parameter TheRanchBTCBullsCommunity.royaltyInfo(uint256,uint256).
    ↪ _salePrice (contracts/TheRanchBTCBullsCommunity.sol#845) is not
    ↪ in mixedCase
Parameter TheRanchBTCBullsCommunity.setBaseURI(string)._newBaseURI (
    ↪ contracts/TheRanchBTCBullsCommunity.sol#853) is not in mixedCase
Parameter TheRanchBTCBullsCommunity.setBaseExtension(string).
    ↪ _newBaseExtension (contracts/TheRanchBTCBullsCommunity.sol#857)
    ↪ is not in mixedCase
```

```
Parameter TheRanchBTCBullsCommunity.setPauseStatus(bool)._paused (
    ↪ contracts/TheRanchBTCBullsCommunity.sol#865) is not in mixedCase
Parameter TheRanchBTCBullsCommunity.setCoreTeamAddresses(address,address
    ↪ )._coreTeam_1 (contracts/TheRanchBTCBullsCommunity.sol#880) is
    ↪ not in mixedCase
Parameter TheRanchBTCBullsCommunity.setCoreTeamAddresses(address,address
    ↪ )._coreTeam_2 (contracts/TheRanchBTCBullsCommunity.sol#880) is
    ↪ not in mixedCase
Parameter TheRanchBTCBullsCommunity.setSafeAddresses(address,address).
    ↪ _hostingSafe (contracts/TheRanchBTCBullsCommunity.sol#886) is not
    ↪  in mixedCase
Parameter TheRanchBTCBullsCommunity.setSafeAddresses(address,address).
    ↪ _btcMinersSafe (contracts/TheRanchBTCBullsCommunity.sol#886) is
    ↪ not in mixedCase
Parameter TheRanchBTCBullsCommunity.setMintingCost(uint256)._price (
    ↪ contracts/TheRanchBTCBullsCommunity.sol#892) is not in mixedCase
Parameter TheRanchBTCBullsCommunity.setUsdcTokenAddress(address).
    ↪ _address (contracts/TheRanchBTCBullsCommunity.sol#897) is not in
    ↪ mixedCase
Parameter TheRanchBTCBullsCommunity.setUsdcTokenDecimals(uint256).
    ↪ _decimals (contracts/TheRanchBTCBullsCommunity.sol#902) is not in
    ↪  mixedCase
Parameter TheRanchBTCBullsCommunity.setWbtcTokenAddress(address).
    ↪ _address (contracts/TheRanchBTCBullsCommunity.sol#906) is not in
    ↪ mixedCase
Parameter TheRanchBTCBullsCommunity.setWbtcTokenDecimals(uint256).
    ↪ _decimals (contracts/TheRanchBTCBullsCommunity.sol#911) is not in
    ↪  mixedCase
Parameter TheRanchBTCBullsCommunity.blacklistMalicious(address,bool).
    ↪ _address (contracts/TheRanchBTCBullsCommunity.sol#915) is not in
    ↪ mixedCase
Parameter TheRanchBTCBullsCommunity.setEcosystemRole(address,bool).
    ↪ _address (contracts/TheRanchBTCBullsCommunity.sol#919) is not in
    ↪ mixedCase
Parameter TheRanchBTCBullsCommunity.setDefenderRole(address,bool).
    ↪ _address (contracts/TheRanchBTCBullsCommunity.sol#923) is not in
    ↪ mixedCase
```

```
Parameter TheRanchBTCBullsCommunity.setChainlinkVrfRole(address,bool).
    ↪ _address (contracts/TheRanchBTCBullsCommunity.sol#927) is not in
    ↪ mixedCase
Parameter TheRanchBTCBullsCommunity.setMonthlyMaintenanceFeePerNFT(
    ↪ uint256)._monthly_maint_fee_per_nft (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#931) is not in mixedCase
Parameter TheRanchBTCBullsCommunity.setStockYardInfo(uint256,uint256,
    ↪ uint256)._stockyardNumber (contracts/TheRanchBTCBullsCommunity.
    ↪ sol#935) is not in mixedCase
Parameter TheRanchBTCBullsCommunity.setStockYardInfo(uint256,uint256,
    ↪ uint256)._startingIndex (contracts/TheRanchBTCBullsCommunity.sol
    ↪ #935) is not in mixedCase
Parameter TheRanchBTCBullsCommunity.setStockYardInfo(uint256,uint256,
    ↪ uint256)._endingIndex (contracts/TheRanchBTCBullsCommunity.sol
    ↪ #935) is not in mixedCase
Variable TheRanchBTCBullsCommunity.coreTeam_1 (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#61) is not in mixedCase
Variable TheRanchBTCBullsCommunity.coreTeam_2 (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#62) is not in mixedCase
Constant TheRanchBTCBullsCommunity.maxSupply (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#69) is not in
    ↪ UPPER_CASE_WITH_UNDERSCORES
Variable TheRanchBTCBullsCommunity.USDCRewardsBalance (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#88) is not in mixedCase
Modifier TheRanchBTCBullsCommunity.ADMIN_OR_DEFENDER() (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#153-156) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (node_modules/@openzeppelin
    ↪ /contracts-upgradeable/access/OwnableUpgradeable.sol#29-31) is
    ↪ not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol
    ↪ #33-35) is not in mixedCase
Variable OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/access/OwnableUpgradeable.sol#94) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init() (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/security/
    ↪ ReentrancyGuardUpgradeable.sol#40-42) is not in mixedCase
```

```
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init_unchained() (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/security/
    ↪ ReentrancyGuardUpgradeable.sol#44-46) is not in mixedCase
Variable ReentrancyGuardUpgradeable.__gap (node_modules/@openzeppelin/
    ↪ contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#74)
    ↪  is not in mixedCase
Function IERC20PermitUpgradeable.DOMAIN_SEPARATOR() (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-
    ↪ IERC20PermitUpgradeable.sol#59) is not in mixedCase
Function ERC721Upgradeable.__ERC721_init(string,string) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/token/ERC721/
    ↪ ERC721Upgradeable.sol#45-47) is not in mixedCase
Function ERC721Upgradeable.__ERC721_init_unchained(string,string) (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/
    ↪ ERC721Upgradeable.sol#49-52) is not in mixedCase
Variable ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/token/ERC721/ERC721Upgradeable.sol#465) is not in
    ↪ mixedCase
Function ERC721EnumerableUpgradeable.__ERC721Enumerable_init() (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/
    ↪ extensions/ERC721EnumerableUpgradeable.sol#16-17) is not in
    ↪ mixedCase
Function ERC721EnumerableUpgradeable.__ERC721Enumerable_init_unchained()
    ↪  (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/
    ↪ extensions/ERC721EnumerableUpgradeable.sol#19-20) is not in
    ↪ mixedCase
Variable ERC721EnumerableUpgradeable.__gap (node_modules/@openzeppelin/
    ↪ contracts-upgradeable/token/ERC721/extensions/
    ↪ ERC721EnumerableUpgradeable.sol#175) is not in mixedCase
Function ContextUpgradeable.__Context_init() (node_modules/@openzeppelin
    ↪ /contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is not
    ↪  in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol
    ↪ #21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/utils/ContextUpgradeable.sol#36) is not in mixedCase
```

```
Function ERC165Upgradeable.__ERC165_init() (node_modules/@openzeppelin/
    ↪ contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol
    ↪ #24-25) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init_unchained() (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/introspection/
    ↪ ERC165Upgradeable.sol#27-28) is not in mixedCase
Variable ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/utils/introspection/ERC165Upgradeable.sol#41) is not
    ↪ in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions

Redundant expression "_tokenId (contracts/TheRanchBTCBullsCommunity.sol
    ↪ #846)" inTheRanchBTCBullsCommunity (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#40-942)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #redundant-statements

Variable TheRanchBTCBullsCommunity.setCoreTeamAddresses(address,address)
    ↪ ._coreTeam_1 (contracts/TheRanchBTCBullsCommunity.sol#880) is too
    ↪  similar to TheRanchBTCBullsCommunity.setCoreTeamAddresses(
    ↪ address,address)._coreTeam_2 (contracts/TheRanchBTCBullsCommunity
    ↪ .sol#880)
Variable TheRanchBTCBullsCommunity.
    ↪ setPayPerNftForTheMonthAndCurrentRewardingDate(uint256,uint256).
    ↪ coreTeam_1_amt (contracts/TheRanchBTCBullsCommunity.sol#353) is
    ↪ too similar to TheRanchBTCBullsCommunity.
    ↪ setPayPerNftForTheMonthAndCurrentRewardingDate(uint256,uint256).
    ↪ coreTeam_2_amt (contracts/TheRanchBTCBullsCommunity.sol#354)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #variable-names-are-too-similar

ReentrancyGuardUpgradeable.__gap (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/security/ReentrancyGuardUpgradeable.sol#74) is never
    ↪ used in TheRanchBTCBullsCommunity (contracts/
    ↪ TheRanchBTCBullsCommunity.sol#40-942)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-state-variable
```

```
initialize() should be declared external:
        - TheRanchBTCBullsCommunity.initialize() (contracts/
          ↪ TheRanchBTCBullsCommunity.sol#216-228)
mint(uint256,bool) should be declared external:
        - TheRanchBTCBullsCommunity.mint(uint256,bool) (contracts/
          ↪ TheRanchBTCBullsCommunity.sol#241-301)
setPayPerNftForTheMonthAndCurrentRewardingDate(uint256,uint256) should
    ↪ be declared external:
        - TheRanchBTCBullsCommunity.
          ↪ setPayPerNftForTheMonthAndCurrentRewardingDate(uint256,
          ↪ uint256) (contracts/TheRanchBTCBullsCommunity.sol#340-367)
rewardBulls(uint256) should be declared external:
        - TheRanchBTCBullsCommunity.rewardBulls(uint256) (contracts/
          ↪ TheRanchBTCBullsCommunity.sol#406-477)
getLiquidatedArrayLength() should be declared external:
        - TheRanchBTCBullsCommunity.getLiquidatedArrayLength() (contracts
          ↪ /TheRanchBTCBullsCommunity.sol#529-531)
setPartnerAddress(address) should be declared external:
        - TheRanchBTCBullsCommunity.setPartnerAddress(address) (contracts
          ↪ /TheRanchBTCBullsCommunity.sol#655-670)
fund() should be declared external:
        - TheRanchBTCBullsCommunity.fund() (contracts/
          ↪ TheRanchBTCBullsCommunity.sol#673)
getRewardAddressesLength() should be declared external:
        - TheRanchBTCBullsCommunity.getRewardAddressesLength() (contracts
          ↪ /TheRanchBTCBullsCommunity.sol#749-751)
getMaintenanceFeeBalanceForAddress() should be declared external:
        - TheRanchBTCBullsCommunity.getMaintenanceFeeBalanceForAddress()
          ↪ (contracts/TheRanchBTCBullsCommunity.sol#753-755)
getMaintenanceFeeStandingForAddress() should be declared external:
        - TheRanchBTCBullsCommunity.getMaintenanceFeeStandingForAddress()
          ↪  (contracts/TheRanchBTCBullsCommunity.sol#757-759)
getWbtcBalanceForAddress() should be declared external:
        - TheRanchBTCBullsCommunity.getWbtcBalanceForAddress() (contracts
          ↪ /TheRanchBTCBullsCommunity.sol#762-764)
getUsdcBalanceForAddress() should be declared external:
```

```
        - TheRanchBTCBullsCommunity.getUsdcBalanceForAddress() (contracts
            ↪ /TheRanchBTCBullsCommunity.sol#767-769)
getPartnerNetworkTeamCount() should be declared external:
        - TheRanchBTCBullsCommunity.getPartnerNetworkTeamCount() (
            ↪ contracts/TheRanchBTCBullsCommunity.sol#774-776)
getAreTheyOnMyPartnerNetworkTeam(address) should be declared external:
        - TheRanchBTCBullsCommunity.getAreTheyOnMyPartnerNetworkTeam(
            ↪ address) (contracts/TheRanchBTCBullsCommunity.sol#781-786)
getRafflePlayer(uint256) should be declared external:
        - TheRanchBTCBullsCommunity.getRafflePlayer(uint256) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#788-790)
getHaveTheyMintedBefore(address) should be declared external:
        - TheRanchBTCBullsCommunity.getHaveTheyMintedBefore(address) (
            ↪ contracts/TheRanchBTCBullsCommunity.sol#795-800)
getMintCountForAddress(address) should be declared external:
        - TheRanchBTCBullsCommunity.getMintCountForAddress(address) (
            ↪ contracts/TheRanchBTCBullsCommunity.sol#802-804)
getNumberOfRafflePlayers() should be declared external:
        - TheRanchBTCBullsCommunity.getNumberOfRafflePlayers() (contracts
            ↪ /TheRanchBTCBullsCommunity.sol#820-822)
getBlacklistedStatus(address) should be declared external:
        - TheRanchBTCBullsCommunity.getBlacklistedStatus(address) (
            ↪ contracts/TheRanchBTCBullsCommunity.sol#824-826)
tokenURI(uint256) should be declared external:
        - ERC721Upgradeable.tokenURI(uint256) (node_modules/@openzeppelin
            ↪ /contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol
            ↪ #98-103)
        - TheRanchBTCBullsCommunity.tokenURI(uint256) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#833-837)
setBaseURI(string) should be declared external:
        - TheRanchBTCBullsCommunity.setBaseURI(string) (contracts/
            ↪ TheRanchBTCBullsCommunity.sol#853-855)
setUsdcTokenAddress(address) should be declared external:
        - TheRanchBTCBullsCommunity.setUsdcTokenAddress(address) (
            ↪ contracts/TheRanchBTCBullsCommunity.sol#897-900)
setUsdcTokenDecimals(uint256) should be declared external:
        - TheRanchBTCBullsCommunity.setUsdcTokenDecimals(uint256) (
            ↪ contracts/TheRanchBTCBullsCommunity.sol#902-904)
```

```
setWbtcTokenAddress(address) should be declared external:
	- TheRanchBTCBullsCommunity.setWbtcTokenAddress(address) (
	  ↪ contracts/TheRanchBTCBullsCommunity.sol#906-909)
setWbtcTokenDecimals(uint256) should be declared external:
	- TheRanchBTCBullsCommunity.setWbtcTokenDecimals(uint256) (
	  ↪ contracts/TheRanchBTCBullsCommunity.sol#911-913)
setStockYardInfo(uint256,uint256,uint256) should be declared external:
	- TheRanchBTCBullsCommunity.setStockYardInfo(uint256,uint256,
	  ↪ uint256) (contracts/TheRanchBTCBullsCommunity.sol#935-941)
renounceOwnership() should be declared external:
	- OwnableUpgradeable.renounceOwnership() (node_modules/
	  ↪ @openzeppelin/contracts-upgradeable/access/
	  ↪ OwnableUpgradeable.sol#66-68)
transferOwnership(address) should be declared external:
	- OwnableUpgradeable.transferOwnership(address) (node_modules/
	  ↪ @openzeppelin/contracts-upgradeable/access/
	  ↪ OwnableUpgradeable.sol#74-77)
name() should be declared external:
	- ERC721Upgradeable.name() (node_modules/@openzeppelin/contracts-
	  ↪ upgradeable/token/ERC721/ERC721Upgradeable.sol#84-86)
symbol() should be declared external:
	- ERC721Upgradeable.symbol() (node_modules/@openzeppelin/
	  ↪ contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol
	  ↪ #91-93)
approve(address,uint256) should be declared external:
	- ERC721Upgradeable.approve(address,uint256) (node_modules/
	  ↪ @openzeppelin/contracts-upgradeable/token/ERC721/
	  ↪ ERC721Upgradeable.sol#117-127)
setApprovalForAll(address,bool) should be declared external:
	- ERC721Upgradeable.setApprovalForAll(address,bool) (node_modules
	  ↪ /@openzeppelin/contracts-upgradeable/token/ERC721/
	  ↪ ERC721Upgradeable.sol#141-143)
transferFrom(address,address,uint256) should be declared external:
	- ERC721Upgradeable.transferFrom(address,address,uint256) (
	  ↪ node_modules/@openzeppelin/contracts-upgradeable/token/
	  ↪ ERC721/ERC721Upgradeable.sol#155-164)
safeTransferFrom(address,address,uint256) should be declared external:
```

```
        - ERC721Upgradeable.safeTransferFrom(address,address,uint256) (
            ↪ node_modules/@openzeppelin/contracts-upgradeable/token/
            ↪ ERC721/ERC721Upgradeable.sol#169-175)
tokenByIndex(uint256) should be declared external:
        - ERC721EnumerableUpgradeable.tokenByIndex(uint256) (node_modules
            ↪ /@openzeppelin/contracts-upgradeable/token/ERC721/
            ↪ extensions/ERC721EnumerableUpgradeable.sol#58-61)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #public-function-that-could-be-declared-external
contracts/TheRanchBTCBullsCommunity.sol analyzed (20 contracts with 78
    ↪ detectors), 205 result(s) found
```

Conclusion:

Most of the vulnerabilities found by the analysis have already been addressed by the smart contract code review.

7 Conclusion

In this audit, we examined the design and implementation of THE RANCH contract and discovered several issues of varying severity. DefiBulls team addressed 7 issues raised in the initial report and implemented the necessary fixes, while classifying the rest as a risk with low-probability of occurrence. Shellboxes' auditors advised DefiBulls Team to maintain a high level of vigilance and to keep those findings in mind in order to avoid any future complications.

8 Disclaimer

Shellboxes reports should not be construed as "endorsements" or "disapprovals" of particular teams or projects. These reports do not reflect the economics or value of any "product" or "asset" produced by any team or project that engages Shellboxes to do a security evaluation, nor should they be regarded as such. Shellboxes Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the examined technology, nor do they provide any indication of the technology's proprietors, business model, business or legal compliance. Shellboxes Reports should not be used in any way to decide whether to invest in or take part in a certain project. These reports don't offer any kind of investing advice and shouldn't be used that way. Shellboxes Reports are the result of a thorough auditing process designed to assist our clients in improving the quality of their code while lowering the significant risk posed by blockchain technology. According to Shellboxes, each business and person is in charge of their own due diligence and ongoing security. Shellboxes does not guarantee the security or functionality of the technology we agree to research; instead, our purpose is to assist in limiting the attack vectors and the high degree of variation associated with using new and evolving technologies.

For a Contract Audit, contact us at contact@shellboxes.com