RCI Smart Contract Final Audit Report

Project Synopsis

Project Name	**RCI**
Platform	Ethereum, Solidity
Github Repo	https://github.com/rocketcapital-ai/tournament-contract/tree/cont-dev
Deployed Contract	Not Deployed
Total Duration	15 Days
Timeline of Audit	**12th September 2021 to 15th September 2021**

Contract Details

Total Contract(s)	5
Name of Contract(s)	Child.sol, Token.sol, Multisig.sol, Competition.sol, CompetitionStorage.sol, Registry.sol
Language	Solidity
Commit Hash	***26ea641b959a17b0a987f429a50d7f0fecde37ad***

Contract Vulnerabilities Synopsis

Issues	Open Issues	Closed Issues
Critical Severity	0	0
Medium Severity	0	2
Low Severity	0	7
Information	0	2
Total Found	0	11

Detailed Results

The contract has gone through several stages of the audit procedure that includes structural analysis, automated testing, manual code review, etc.

All the issues have been explained and discussed in detail below. Along with the explanation of the issue found during the audit, the recommended way to overcome the issue or improve the code quality has also been mentioned.

A. Contract Name: Competition.sol, CompetitionStorage

High Severity Issues

None Found

Medium Severity Issues

A.1 Function visibility issue found in *updateChallengeAndTournamentScores() function*

Line no -387-389, 394-396

Status: CLOSED

Explanation:

The **Competition** contract includes a function called ***updateChallengeAndTournamentScores*** to store the challenge and tournament scores of participants on-chain.

As per the current architecture of the contract, most of the ***onlyAdmin*** functions are usually divided into two parts where the first one is marked external which allows the admin to access the function. While the 2nd part with the actual function logic is made private, thus only accessible by its respective external function.

However, the same pattern wasn't found with the ***updateChallengeAndTournamentScores*** function as the function with the logic, in this case, is assigned **Public visibility (***Line 394 to 396***),** thus making both functions with similar names accessible from outside the contract.

Moreover, while the function with external visibility reads the ***_challengeCounter*** directly from the contract, the **public function** demands it to be passed by the admin which might not be a very effective mechanism.

```
386
387     function updateChallengeAndTournamentScores(address[] calldata participants, uint256[] calldata challengeScores, uint256
388     external override
389     returns (bool success)
390     {
391         success = updateChallengeAndTournamentScores(_challengeCounter, participants, challengeScores, tournamentScores);
392     }
393
394     function updateChallengeAndTournamentScores(uint32 challengeNumber, address[] calldata participants, uint256[] calldata
395     public override onlyAdmin
396     returns (bool success)
397     {
```

Recommendation:

If the above-mentioned scenario is not intended, the function visibility of the function should be updated accordingly.

A.2 Violation of Check_Effects_Interaction Pattern in the Withdraw function

Line no - 689-700

Status: CLOSED

Explanation:

The **Competition** contract includes function**, *sponsor(),*** that update some of the very imperative state variables of the contract after the external calls are being made.

An external call within a function technically shifts the control flow of the contract to another contract for a particular period of time. Therefore, as per the Solidity Guidelines, any modification of the state variables in the base contract must be performed before executing the external call.
Although in this case, the call is being made to the native token contract itself, it's imperative to not violate the best security practices.

The following function in the contract update the state variables after making an external call at the lines mentioned below:

- ***sponsor*** *at Line* ***696***

```
689     function sponsor(uint256 amountToken)
690     external override
691     returns (bool success)
692     {
693         require(_challenges[_challengeCounter].phase == 4, "Competition -
694         _token.transferFrom(msg.sender, address(this), amountToken);
695         uint256 currentCompPoolAmt = _competitionPool;
696         _competitionPool = currentCompPoolAmt + amountToken;
697         success = true;
```

Recommendation:

Check Effects Interaction Pattern must be followed while implementing external calls in a function.

Low Severity Issues

A.3 Adequate use of Return Value of an External Call is was not found

Line no - 80, 694

Status: CLOSED

Explanation:

The external calls made in the above-mentioned lines do return a boolean value that indicates whether or not the external call made was successful.

These boolean return values can be used in the function as a check to ensure that the further execution of the function is only allowed if the external is successfully made. However, the **Competition** contract never uses these return values throughout the contract.

Recommendation:

Effective use of all the return values from external calls must be ensured within the contract.

A.4 External Visibility should be preferred

Status: CLOSED

Explanation:

Functions that are never called throughout the contract should be marked as ***external*** visibility instead of ***public*** visibility.
This will effectively result in Gas Optimization as well.

During the automated testing of the ***Competition*** contract, it was found that the following functions could be marked as **external** within the contract:

- ***updateSubmission()***
- ***updateResults()***

Recommendation:

If the **PUBLIC** visibility of the above-mentioned functions is not intended, then the **EXTERNAL** Visibility keyword should be preferred.

B. Contract Name: Multisig

High Severity Issues
None Found

Medium Severity Issues
None Found

Low Severity Issues
B.1 Redundant Require Statement found in *removeOwner* function

Status: CLOSED

Line - 134

Explanation:

As per the current architecture of the removeOwner function, it was found that it contains a ***notNull()*** modifier which ensures that the address of the owner is not a **zero address.**

However, this validation has already been performed while adding a particular owner, in the ***addOwner function*** at *Line 120.*

This makes the ***notNull modifier*** in the ***removeOwner function*** redundant and badly affects the gas optimization of the function.

```
128     /// @dev Allows to remove an owner. Transaction has
129     /// @param owner Address of owner.
130     function removeOwner(address owner)
131         public
132         onlyWallet
133         ownerExists(owner)
134         notNull(owner)
135         validRequirement(owners.length - 1, required)
136     {
```

Recommendation:

Redundant require statements and validations should be avoided.

B.2 Absence of Error messages in Require Statements

Line no - 49-95, 106

Status: CLOSED

Description:

The **Multisig** contract includes a few functions(*at the above-mentioned lines*) that don't contain any error message in the **require** statement.

While this makes it troublesome to detect the reason behind a particular function revert, it also reduces the readability of the code.

Recommendation:

Error Messages must be included in every require statement in the contract

B.3 External Visibility should be preferred

Status: CLOSED

Explanation:

Functions that are never called throughout the contract should be marked as ***external*** visibility instead of ***public*** visibility.
This will effectively result in Gas Optimization as well.

During the automated testing of the ***MultiSig*** contract, it was found that the following functions could be marked as **external** within the contract:

- ***addOwner***
- ***removeOwner***
- ***replaceOwner***
- ***submitTransaction***
- ***revokeConfirmation***
- ***getConfirmationCount***
- ***getTransactionCount***
- ***getOwners***
- ***getConfirmations***
- ***getTransactionIds***

Recommendation:

If the **PUBLIC** visibility of the above-mentioned functions is not intended, then the **EXTERNAL** Visibility keyword should be preferred.

Informational

B.4 Internal function *isContract()* is never used within the contract

Status: CLOSED

Line no 427-439

Explanation:

The Multisig Contract includes an internal function called ***isContract()*** at the above-mentioned line.

However, the function is never used as the contract uses the Address library imported in the contract.

Recommendation:

Unnecessary state variables and functions must be removed.

B.5 Commented codes must be wiped out before deployment

Explanation

The Multisig contract includes quite a few commented codes**.** This affects the readability of the code.

Recommendation:

If these instances of code are not required in the current version of the contract, then the commented codes must be removed before deployment.

C. Contract Name: Token. sol

Low Severity Issues

C.1 Absence of Zero Address Validation

Line no- 78, 85

Status: CLOSED

Description:

The **Token** Contract includes quite a function called ***authorizeCompetition***, which updates an imperative mapping, i.e., ***_authorizedCompetitions,*** in the contract.

```
78      function authorizeCompetition(address competitionAddress)
79      external
80      onlyAdmin
81      {
82          _authorizedCompetitions[competitionAddress] = true;
83
84          emit CompetitionAuthorized(competitionAddress);
85      }
```

However, during the automated testing of the contract, it was found that no Zero Address validation is implemented before updating the address of the mapping.

Although the function has already been assigned an **onlyOwner** modifier, keeping in mind the immutable nature of the smart contract, it's imperative to implement input validations in the function.

Recommendation:

A **require** statement should be included in such functions to ensure no invalid address is passed in the arguments.

D. Contract Name: Registry.sol

High Severity Issues

None Found

Medium Severity Issues

None Found

Low Severity Issues

D.1 Absence of Input Validation found in few functions

Line no - *33-41, 71-79*

Status: CLOSED

Explanation:

The registry contract includes functions like ***registerNewCompetition*** and ***registerNewExtension*** that doesn't involve any input validation for the following arguments:

- ***a. competitionAddress***
- ***b. rulesLocation***
- ***c. extensionAddress***
- ***d. informationLocation***

It's imperative to implement adequate input validation to avoid unwanted behavior during contract execution.

***Recommendation*:**

Effective input validations should be included.

E. Contract Name: ChildToken.sol

No Issues Found