CERTIK

Security Assessment

Chair

May 23rd, 2021

Table of Contents

Summary

This report has been prepared for Chair smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Addtionally, this audit is based on a premise that all external contracts were implemented safely.

The security assessment resulted in 9 findings that ranged from minor to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Chair
Platform	BSC
Language	Solidity
Codebase	https://github.com/ChairOfficial/smart-contract
Commits	5178b2a9bf255d46aef3f6e364c47de4957493f3

Audit Summary

Delivery Date	May 23, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Total Issues	9
Critical	0
Major	0
Medium	0
Minor	4
Informational	5
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
NMC	NftMarket.sol	da73a5247411d85659226cfe72bb44f02aa5bd756ba060b05f3a0fb455d06790
CKP	erc20.sol	34103145c31a6c8d501dc688962abafe2582615768a739d326ac825b21aebc0b
ERC	erc721.sol	1b315317ece824f000fad39838f09a01483abd4d59860bd40dff87daf4680cb5

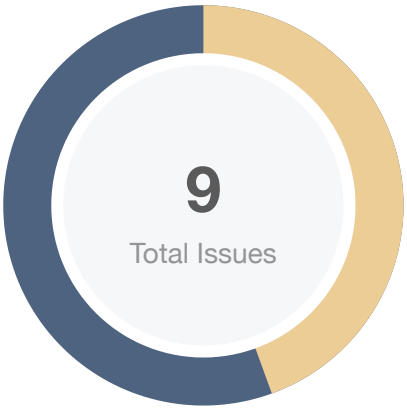
Centralization Roles

The Chair smart contract introduces an authorization.

Owner:

[NftMarket.sol]: transferOwnership(): transfer ownership to `newOwner`;
recommend(): set a nft to be recommend;
cancelRecommend(): set a nft to be not recommended;
recoveryEth(): get all `BNB` from the contract.

Findings



Critical	**0** (0.00%)
Major	**0** (0.00%)
Medium	**0** (0.00%)
Minor	**4** (44.44%)
Informational	**5** (55.56%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
ERC-01	Missing Zero Address Validation	Logical Issue	Informational	Resolved
NMC-01	Missing Zero Address Validation	Logical Issue	Informational	Resolved
NMC-02	Proper Usage of "public" And "external" Type	Gas Optimization	Informational	Resolved
NMC-03	Lack Of Verification For `msg.value`	Logical Issue	Informational	Declined
NMC-04	Centralization Permission of `recoveryEth` Function	**Centralization / Privilege**	**Minor**	**Acknowledged**
NMC-05	Incorrect Calculation In The `enterBidForNft` Function	Logical Issue	Minor	Resolved
NMC-06	Divide Before Multiply	Mathematical Operations	Informational	Resolved
NMC-07	Incorrect Calculation In The `deal` Function	Logical Issue	Minor	Resolved
NMC-08	Security Optimization	Logical Issue	Minor	Resolved

ERC-01 | Missing Zero Address Validation

Category	Severity	Location	Status
Logical Issue	Informational	erc721.sol: 460	Resolved

Description

Addresses should be checked before assignment to make sure they are not zero addresses. This suggestion is not limited to these codes but also applies to other similar codes.

Recommendation

Consider adding a check like below:

```
1 require(player != address(0), "player address cannot be 0");
```

Alleviation

The team heeded our advice and changed related code. Code change was applied in commit a664dd0796721f1128c8729a4ce3dad878953681.

NMC-01 | Missing Zero Address Validation

Category	Severity	Location	Status
Logical Issue	Informational	NftMarket.sol: 116~118, 42	Resolved

Description

Addresses should be checked before assignment to make sure they are not zero addresses. This suggestion is not limited to these codes but also applies to other similar codes.

Recommendation

Consider adding a check like below:

```
require(_nftAsset != address(0), "_nftAsset address cannot be 0");
require(_abcToken != address(0), "_abcToken address cannot be 0");
require(_revenueRecipient != address(0), "_revenueRecipient address cannot be 0");
```

Alleviation

The team heeded our advice and changed related code. Code change was applied in commit a664dd0796721f1128c8729a4ce3dad878953681.

NMC-02 | Proper Usage of "public" And "external" Type

Category	Severity	Location	Status
Gas Optimization	Informational	NftMarket.sol: 228	Resolved

Description

`public` functions that are never called by the contract could be declared `external`.

Recommendation

Consider using the `external` attribute for functions never called from the contract.

Alleviation

The team heeded our advice and changed related code. Code change was applied in commit a664dd0796721f1128c8729a4ce3dad878953681.

NMC-03 | Lack Of Verification For `msg.value`

Category	Severity	Location	Status
Logical Issue	Informational	NftMarket.sol: 256	Declined

Description

When users use `eth` to bid, `msg.value` should equal to `amount`.

Recommendation

Add verification, verify that `msg.value` is equal to `amount`. For example:

```
if (offer.paymentToken != address(0)) {
            ...
        } else {
            require(msg.value==amount,"msg.value must equal to amount!");
            ...
        }
```

Alleviation

No alleviation.

NMC-04 | Centralization Permission of `recoveryEth` Function

Category	Severity	Location	Status
Centralization / Privilege	Minor	NftMarket.sol: 369	Acknowledged

Description

What is the purpose of designing this function, can you provide a detailed introduction please?

Recommendation

We advise the client to carefully manage the `_owner` role's accounts' private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

Alleviation

The team responded that this function is used to transfer `BNB` out in case of users transferred wrong amount of `BNB` to this contract or `BNB` of contract can not be withdrawn.

NMC-05 | Incorrect Calculation In The `enterBidForNft` Function

Category	Severity	Location	Status
Logical Issue	Minor	NftMarket.sol: 254, 266	Resolved

Description

`amount` represents the additional amount each time, after the bid is successful, the `bid.value` should be updated to the whole `offerBalance`.

Recommendation

Use `amount + offerBalances[tokenID][msg.sender]` instead of `amount`. For example:

```
1  nftBids[tokenID] = Bid(tokenID, msg.sender, amount + offerBalances[tokenID]
[msg.sender]);
```

Alleviation

The team heeded our advice and changed related code. Code change was applied in commit a664dd0796721f1128c8729a4ce3dad878953681.

NMC-06 | Divide Before Multiply

Category	Severity	Location	Status
Mathematical Operations	Informational	NftMarket.sol: 292~295, 323~326	Resolved

Description

Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision.

Recommendation

Consider ordering multiplication before division.For example:

```
    uint256 tempC =
                bid.value * offer.reward *
                offerBalances[tokenID][bidders[tokenID][i]] /
                totalBid / 100;
```

Alleviation

The team heeded our advice and changed related code. Code change was applied in commit a664dd0796721f1128c8729a4ce3dad878953681.

NMC-07 | Incorrect Calculation In The `deal` Function

Category	Severity	Location	Status
Logical Issue	Minor	NftMarket.sol: 292, 323	Resolved

Description

When processing the auction results, it is obviously wrong to refund the money to all users who participated in the bidding. Should only refund the money to the users who failed.

Recommendation

Add a judgment condition, only refund the money to the users who failed. For example:

```
for (uint256 i = 0; i < bidders[tokenID].length; i++) {
                    if(bid.bidder!=bidders[tokenID][i]){
                              uint256 tempC =
                        (((bid.value * offer.reward) / 100) *
                            offerBalances[tokenID][bidders[tokenID][i]]) /
                            totalBid;
                    payable(bidders[tokenID][i]).transfer(tempC);
                    share3 += tempC;
                    payable(bidders[tokenID][i]).transfer(
                        offerBalances[tokenID][bidders[tokenID][i]]
                    );
                    offerBalances[tokenID][bidders[tokenID][i]] = 0;
                    delete bade[tokenID][bidders[tokenID][i]];
                    }
                }
```

Alleviation

The team heeded our advice and changed related code. Code change was applied in commit a664dd0796721f1128c8729a4ce3dad878953681.

NMC-08 | Security Optimization

Category	Severity	Location	Status
Logical Issue	Minor	NftMarket.sol: 145, 173, 184, 228, 272	Resolved

Description

Please make sure whether the `offer.paymentToken` and `abcToken` are safe. If not, we advice using `ReentrancyGuard` to avoid reentrancy attack.

Recommendation

Consider adding a reentrancy lock.

Alleviation

The team heeded our advice and added a reentrancy lock. Code change was applied in commit a664dd0796721f1128c8729a4ce3dad878953681.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.