Security Assessment

LoserChick

Jun 13th, 2021

Table of Contents

Summary

This report has been prepared for LoserChick 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.

The security assessment resulted in findings that ranged from critical 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	LoserChick
Description	Loser chick is a NFT game project in Defi field which is a world of chicks built on BSC.
Platform	BSC
Language	Solidity
Codebase	https://github.com/Loserchick/loserchick_contracts
Commit	3d33c42a1b9cfddac03293d6cc28ed0a8ba9c23e 713c985e0bc55c8130c2a64cf0fd1cac3ac76ebb

Audit Summary

Delivery Date	Jun 13, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

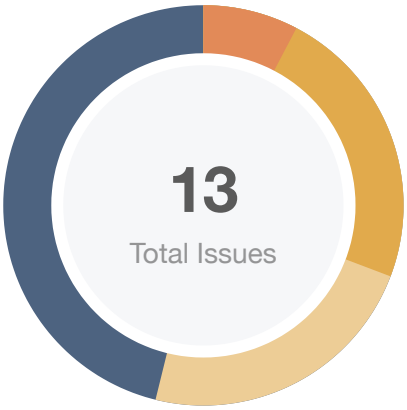
Vulnerability Summary

Total Issues	13
Critical	0
Major	1
Medium	3
Minor	3
Informational	6
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
CMC	ChickMining.sol	4c05d300ce566b2bd183bddcfa12d273de29adde5f693d0f3cbd768fd5466fa1
CTC	ChickToken.sol	10b1259dd945119d666fb0ec47b355284efd4a4ecc99a536f9d2e0ebb8fa162f
ETC	EggToken.sol	268aa3b929c0d15e22f4311e97c6c2aef69485ace37524618b453f566593c3b3
LCN	LoserChickNFT.sol	ba6899f24afab3ad6c5567ca2252f8d6cc4fe132d9c2b7baee504fc3f7633a1c
OCC	OwnableContract.sol	921df42f84ce6f49b4cb062ee6dea71a9419e6702b32521e79b9cffd32a7c84e
RIC	RandomInterface.sol	0c55aa9630a853901f8f8bdfb7370574ab966c91ea012915e8a9382ef9e0f1f0
SEC	SmashEggs.sol	cf132e65c8b47de15a551d2072e449db54e211b2afb32990190352ffe106c026

Findings



Critical	**0** (0.00%)
Major	**1** (7.69%)
Medium	**3** (23.08%)
Minor	**3** (23.08%)
Informational	**6** (46.15%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
CMC-01	add() Function Not Restricted	Volatile Code	Medium	Acknowledged
CMC-02	Use "busdBaseUnit" instead of "1e18"	Coding Style	Informational	Resolved
CMC-03	Check Effect Interaction Pattern Violated	Logical Issue	Minor	Acknowledged
CTC-01	Inaccurate function name	Logical Issue	Major	Acknowledged
ETC-01	Set Stable Variables with `constant`	Gas Optimization	Informational	Resolved
ETC-02	Comparison with boolean	Gas Optimization	Informational	Acknowledged
ETC-03	Use SafeMath	Mathematical Operations	Minor	Resolved
LCN-01	Set Stable Variables with `immutable`	Gas Optimization	Informational	Resolved
LCN-02	Missing semicolon	Coding Style	Minor	Resolved
RIC-01	Unknown Implementation of RandomInterface.getRandomNumber	Logical Issue	Informational	Acknowledged
SEC-01	Discussion on "chickProbability" value	Logical Issue	Informational	Resolved
SEC-02	Centralized risk in "updateActivityNFT"	**Centralization / Privilege**	**Medium**	**Acknowledged**
SEC-03	Centralized risk in "transferActivityNFT"	**Centralization / Privilege**	**Medium**	**Resolved**

CMC-01 | add() Function Not Restricted

Category	Severity	Location	Status
Volatile Code	Medium	ChickMining.sol: 146	Acknowledged

Description

The comment in line 145, mentioned `// Add a new lp to the pool.`

However, the code is not reflected in the comment behaviors as there isn't any valid restriction on preventing this issue.

The current implementation relies on the owner's trust to avoid repeatedly adding the same LP token to the pool, as the function will only be called by the owner.

Recommendation

Detect whether the given pool for addition is a duplicate of an existing pool. The pool addition is only successful when there is no duplicate. Using mapping of `addresses` -> `booleans`, which can restrict the same address being added twice.

Alleviation

[LoserChick Team]: It is called by admin, we have a solution to avoid repeatedly adding the same LP token to the pool.

CMC-02 | Use "busdBaseUnit" instead of "1e18"

Category	Severity	Location	Status
Coding Style	Informational	ChickMining.sol: 238	Resolved

Description

There is a defined constant variable `busdBaseUnit`, but not used. At the same time, there are some `1e18` constent variables are used.

Recommendation

We recommend using `busdBaseUnit` instead of `1e18` in the contract.

Alleviation

LoserChick team heeded our advise and changed the code in commit `713c985e0bc55c8130c2a64cf0fd1cac3ac76ebb`.

CMC-03 | Check Effect Interaction Pattern Violated

Category	Severity	Location	Status
Logical Issue	Minor	ChickMining.sol: 286	Acknowledged

Description

The order of external call/transfer and storage manipulation must follow the check-effect-interaction pattern.

Recommendation

We advise the client to check if storage manipulation is before the external call/transfer operation. LINK

Alleviation

No Alleviation

CTC-01 | Inaccurate function name

Category	Severity	Location	Status
Logical Issue	Major	ChickToken.sol: 14	Acknowledged

Description

Function `chickSwapCchick` is `public`, it can be called by any address. And it is used to burn the caller's CHICK token, but it is named as a swap method. This name does not match its running logic.

Recommendation

We recommend naming the function accurectly.

Alleviation

[LoserChick Team]: Cchick is the data stored in the centralized database.

ETC-01 | Set Stable Variables with `constant`

Category	Severity	Location	Status
Gas Optimization	Informational	EggToken.sol: 8	Resolved

Description

Variable `MAX_TOTAL_SUPPLY` could be declared `constant`.

Recommendation

We advise declaring `MAX_TOTAL_SUPPLY` constant.

Alleviation

LoserChick team heeded our advise and changed the code in commit `713c985e0bc55c8130c2a64cf0fd1cac3ac76ebb`.

ETC-02 | Comparison with boolean

Category	Severity	Location	Status
Gas Optimization	Informational	EggToken.sol: 67	Acknowledged

Description

The following code performs a comparison with a boolean literal, which can be replaced with the negation of the expression to increase the legibility of the codebase.

```
67      require(claimedOrderId[orderId] == false, "already claimed");
```

Recommendation

We advise that use the expression inside the `require` statement instead of comparison with boolean.

Alleviation

No Alleviation

ETC-03 | Use SafeMath

Category	Severity	Location	Status
Mathematical Operations	Minor	EggToken.sol: 82~85, 110	Resolved

Description

Avoid use the operators such as `+=` in the Solidity contract.

Recommendation

We advise that use related functions of SafeMath like `add()`.

Alleviation

LoserChick team heeded our advise and changed the code in commit `713c985e0bc55c8130c2a64cf0fd1cac3ac76ebb`.

LCN-01 | Set Stable Variables with `immutable`

Category	Severity	Location	Status
Gas Optimization	Informational	LoserChickNFT.sol: 28	Resolved

Description

Variable `maxSupply` is only initialized once in the constructor of the smart contract.

Recommendation

We advise the client to consider adding keyword `immutable` to the `maxSupply` variable.

Alleviation

LoserChick team heeded our advise and changed the code in commit `713c985e0bc55c8130c2a64cf0fd1cac3ac76ebb`.

LCN-02 | Missing semicolon

Category	Severity	Location	Status
Coding Style	Minor	LoserChickNFT.sol: 52	Resolved

Description

Missing semicolon in line 52.

Recommendation

We advise that add a semicolon at the ending of every line code.

Alleviation

LoserChick team heeded our advise and changed the code in commit `713c985e0bc55c8130c2a64cf0fd1cac3ac76ebb`.

RIC-01 | Unknown Implementation of RandomInterface.getRandomNumber

Category	Severity	Location	Status
Logical Issue	Informational	RandomInterface.sol: 5	Acknowledged

Description

Function `getRandomNumber` is used to get a random number for this smart contract. The implementation of `getRandomNumber` is not in the scope of the audit.

Recommendation

We advise that opensource and show how to get the random numbers?

Alleviation

[LoserChick Team]: In order to avoid some advanced players using random numbers to predict the game results, this part of the code is not open source.

SEC-01 | Discussion on "chickProbability" value

Category	Severity	Location	Status
Logical Issue	Informational	SmashEggs.sol: 63~66	Resolved

Description

The comments in line 63 to 66, mentioned some probabilities of winning. However, the constant values are different with these comments. Which group is match the intention of design?

Alleviation

[LoserChick Team]: It just means probability.

SEC-02 | Centralized risk in "updateActivityNFT"

Category	Severity	Location	Status
Centralization / Privilege	**Medium**	SmashEggs.sol: 81	**Acknowledged**

Description

Function `updateActivityNFT` can set any NFT token addresses to `activityNFTAddr` and `activityNFTProbability` by the owner. As result, invocation of `updateActivityNFT` may set the variables as two unknown NFT tokens. However, the project may lose the ability to upgrade if `updateActivityNFT` is removed.

Recommendation

To improve the trustworthiness of this project, any plan to set the `activityNFTAddr` and `activityNFTProbability` should move to the execution queue of the Timelock and also add an `emit event`, and Multisig with community-selected 3-party independent co-signers, and/or DAO with transparent governance with the project's community in the project to manage sensitive role accesses.

Alleviation

[LoserChick Team]: When the probability is 0, the relevant logic is not called.

SEC-03 | Centralized risk in "transferActivityNFT"

Category	Severity	Location	Status
Centralization / Privilege	**Medium**	SmashEggs.sol: 221	**Resolved**

Description

Function `transferActivityNFT` is only called by the admin, and it allows the caller to transfer any count NFT tokens to specified addresses. `transferActivityNFT` has the possibility of being maliciously manipulated by hacker if the account of `admin` was compromised.

Recommendation

We advise the client to carefully manage the project's private key and avoid any potential risks of being hacked. We also advise the client to adopt Timelock with reason delay to allow the admin to transfer the NFT token, Multisig with community-selected 3-party independent co-signers, and/or DAO with transparent governance with the project's community in the project to manage sensitive role accesses.

Alleviation

[LoserChick Team]: After the deployment of the main network, the administrator permissions will be transferred to multiple signings.

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.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

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.