Smart Contract Audit Report

for

Defina

TRUSTLOOK

Version 1.0

Trustlook Blockchain Labs

Email: bd@trustlook.com

Project Overview

Project Name	Ludena
Contract codebase	N/A
Platform	EVM compatible blockchains
Language	Solidity
Submission Time	2022.07.25

Report Overview

Report ID	TBL_20220725_00
Version	1.0
Reviewer	Trustlook Blockchain Labs
Starting Time	2022.07.25
Finished Time	2022.07.30

Disclaimer

Trustlook audit reports do not provide any warranties or guarantees on the vulnerability-free nature of the given smart contracts, nor do they provide any indication of legal compliance. The Trustlook audit process is aiming to reduce the high level risks possibly implemented in the smart contracts before the issuance of audit reports. Trustlook audit reports can be used to improve the code quality of smart contracts and are not able to detect any security issues of smart contracts that will occur in the future. Trustlook audit reports should not be considered as financial investment advice.

About Trustlook Blockchain Labs

Trustlook Blockchain Labs is a leading blockchain security team with a goal of security and vulnerability research on current blockchain ecosystems by offering industry-leading smart contracts auditing services. Please contact us for more information at (https:/www.trustlook.com/services/smart.html) or Email (bd@trustlook.com)

The Trustlook blockchain laboratory has established a complete system test environment and methods.

Black-box Testing	The tester has no knowledge of the system being attacked. The goal is to simulate an external hacking or cyber warfare attack.
White-box Testing	Based on the level of the source code, test the control flow, data flow, nodes, SDK etc. Try to find out the vulnerabilities and bugs.
Gray-box Testing	Use Trustlook customized script tools to do the security testing of code modules, search for the defects if any due to improper structure or improper usage of applications.

Introduction

By reviewing the smart contract’s implementation, this audit report has been prepared to discover potential issues and vulnerabilities of their source code. We outline in the report about our approach to evaluate the potential security risks. Advice to further improve the quality of security or performance is also given in the report.

About Defina

Defina.Finance is a data aggregator specially designed for DeFi and NFT, providing customisable smart contracts to simplify the investment process of DeFi and NFT for users of all levels.

About Methodology

To evaluate the potential vulnerabilities or issues, we go through a checklist of well-known smart contracts related security issues using automatic verification tools and manual review. To discover potential logic weaknesses or project specific implementations, we thoroughly discussed with the team to understand the business model and reduce the risk of unknown vulnerabilities. For any discovered issue, we might test it on our private network to reproduce the issue to prove our findings.

The checklist of items is shown in the following table:

Category	Type ID	Name	Description
Coding Specification	CS-01	ERC Standards	The contract is using ERC standards.
	CS-02	Compiler Version	The compiler version should be specified.
	CS-03	Constructor Mismatch	The constructor syntax is changed with Solidity versions. Need extra attention to make the constructor function right.
	CS-04	Return standard	Following the ERC20 specification, the transfer and approve functions should return a bool value, and a return value code needs to be added.
	CS-05	Address(0) Validation	It is recommended to add the verification of require(_to!=address(0)) to effectively avoid unnecessary loss caused by user misuse or unknown errors.
	CS-06	Unused Variable	Unused variables should be removed.
	CS-07	Untrusted Libraries	The contract should avoid using untrusted libraries, or the libraries need to be thoroughly audited too.
	CS-08	Event Standard	Define and use Event appropriately
	CS-09	Safe Transfer	Using safeTransfer/transfer to send funds instead of send.
	CS-10	Gas Consumption	Optimize the code for better gas consumption.
	CS-11	Deprecated Uses	Avoid using deprecated functions.
	CS-12	Sanity Checks	Sanity checks when setting key parameters in the system
	CS-13	Typo	Typo in comments or code
	CS-14	Fallback Function	Splitting fallback and receive function
	CS-15	Comment Standard	Use clear consistent comments with code semantics
	CS-16	Naming Standard	Use standard method to name functions and variables

Coding Security	SE-01	Integer overflows	Integer overflow or underflow issues.
	SE-02	Reentrancy	Avoid using calls to trade in smart contracts to avoid reentrancy vulnerability.
	SE-03	Transaction Ordering Dependence	Avoid transaction ordering dependence vulnerability.
	SE-04	Tx.origin usage	Avoid using tx.origin for authentication.
	SE-05	Fake recharge	The judgment of the balance and the transfer amount needs to use the “require function”.
	SE-06	Replay	If the contract involves the demands for entrusted management, attention should be paid to the non-reusability of verification to avoid replay attacks.
	SE-07	External call checks	For external contracts, pull instead of push is preferred.
	SE-08	Weak random	The method of generating random numbers on smart contracts requires more considerations.
Additional Security	AS-01	Access control	Well defined access control for functions.
	AS-02	Authentication management	The authentication management is well defined.
	AS-03	Semantic Consistency	Semantics are consistent.
	AS-04	Functionality checks	The functionality is well implemented.
	AS-05	Business logic review	The business model logic is implemented correctly.

The severity level of the issues are described in the following table:

Severity	**Description**
Critical	The issue will result in asset loss or data manipulations.
High	The issue will seriously affect the correctness of the business model.
Medium	The issue is still important to fix but not practical to exploit.
Low	The issue is mostly related to outedate, unused code snippets.
Informational	This issue is mostly related to code style, informational statements and is not mandatory to be fixed.

Audit Results

Here are the audit results of the smart contracts.

Scope

Following files have been scanned by our internal audit tool and manually reviewed and tested by our team:

File names	Sha1
CommonUtils.sol	421c56ebcc280637fb4f0787021cd5ebe1c6d063
NewDefinaCardEventsAndErrors.sol	2e21ec79ce9776eca5365f5715d3d756645e0ca4
NewDefinaCardInterface.sol	3ed0d2e62711390fc297938e35676361cb390133
NewDefinaCardStructs.sol	6793beb4e8b277e922f1d18b61422c1b8573c172
NewDefinaCardV2.sol	4c00fe0d71cc4ed7087918b2c61cd40132fd777a
RandomSend.sol	df4aafed19e677c7b2437f83320eda1da78e2639

Summary

Issue ID	Severity	Location	Type ID	Status
TBL_SCA_001	Medium	NewDefinaCardV2.sol:362	AS-05	Fixed
TBL_SCA_002	Medium	NewDefinaCardV2.sol:342	AS-04	Fixed
TBL_SCA_003	Low	NewDefinaCardV2.sol:226	AS-04	Fixed
TBL_SCA_004	Low	CommonUtils.sol, RandomSend.sol	SE-08	Closed
TBL_SCA_005	Info	NewDefinaCardV2.sol: 111	CS-12	Fixed
TBL_SCA_006	Info	NewDefinaCardV2.sol:148,155,159	CS-10	Fixed
TBL_SCA_007	Info	NewDefinaCardV2.sol: 239	CS-10	Fixed
TBL_SCA_008	Info	NewDefinaCardV2.sol: 329	CS-10	Fixed
TBL_SCA_009	Info	NewDefinaCardEventsAndErrors.sol:10, 13, 14, RandomSend.sol:11, 12	CS-08	Fixed

Details

- ID: TBL_SCA-001
- Severity: Medium
- Type: AS-05 (Business logic review)
- Location: NewDefinaCardV2.sol (362)
- Description:

When two tokens A and B were used to call addMerge(). The merge operation is stored in mergeMap by index of token ID of A. Therefore, only when A token is transferred to a new owner, the merge record can be retrieved in the _beforeTokenTransfer() function and the record will be removed. However, if B was transferred to a new owner, the record will be kept and both A and B in forMerge mapping are true.

It is recommended to check forMerge[B] in function _beforeTokenTransfer() and also remove the merge record in mergeMap if B was transferred. Otherwise, both A and B will be freezed to merge again in future.

- Remediation:

This issue has been fixed.

- ID: TBL_SCA-002
- Severity: Medium
- Type: AS-04 (Functionality checks)
- Location: NewDefinaCardV2.sol (342)
- Description:

The loop in the function toMerge() is aimed at finding a heroId which is not marked as True in the iSOkexMap. The iteration time of the loop could be huge or even infinite in the worst case scenario.

It is recommended to supply a list of hero IDs which are all false in iSOkexMap to CommonUtils.getHeroBySeed() to avoid using the loop.

Line 345 “index = 0;” can also be removed.

- Remediation:

This issue has been fixed.

- ID: TBL_SCA-003
- Severity: Low
- Type: AS-04 (Functionality checks)
- Location: NewDefinaCardV2.sol (226)
- Description:

There is a boundary error in the function nftOwnerClaimCards(). When the tokeId is equal to maxClaimedAmount, the ID is acceptable to be claimed. However, this ID could be minted in the function mintMulti() which should not be claimed by the business logic.

It is recommended to update the validation to be:

"(tokenId >= maxClaimedAmount)"

- Remediation:

This issue has been fixed.

- ID: TBL_SCA-004
- Severity: Low
- Type: SE-08 (Weak Randomness)
- Location: CommonUtils.sol, RandomSend.sol
- Description:

  The random number generators used in these files are predictable.

- Remediation:

  The development team is aware of this and has decided to leave it as is.

- ID: TBL_SCA-005
- Severity: Info
- Type: CS-12 (Sanity Checks)
- Location: NewDefinaCardV2.sol (111)
- Description:

  It is recommended to check that *nftAmount_* is much smaller than *MAX_MINT*.

- Remediation:

  This issue has been fixed.

- ID: TBL_SCA-006
- Severity: Info
- Type: CS-10 (Gas Consumption)
- Location: NewDefinaCardV2.sol (148,155,159)
- Description:

  The *delete* is not necessary and can be removed.

- Remediation:

  This issue has been fixed.

• ID: TBL_SCA-007

• Severity: Info

• Type: CS-10 (Gas Consumption)

• Location: NewDefinaCardV2.sol (239)

• Description:

It is recommended to store heroIdMap[tokenId] in a local variable and use the local variable in the following statements.

• Remediation:

This issue has been fixed.

- ID: TBL_SCA-008
- Severity: Info
- Type: CS-10 (Gas Consumption)
- Location: NewDefinaCardV2.sol (329)
- Description:

  Variable *success* is not needed, but can be replaced with *mergeResult*.

- Remediation:

  This issue has been fixed.

- ID: TBL_SCA-009
- Severity: Info
- Type: CS-08 (Event Standard)
- Location: NewDefinaCardEventsAndErrors.sol (10, 13, 14), RandomSend.sol (11, 12)
- Description:

  It is recommended to index the address.

- Remediation:

  This issue has been fixed.