CERTIK

Security Assessment

Legend

Jun 8th, 2021

Table of Contents

Summary

This report has been prepared for Legend 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	Legend
Platform	Heco
Language	Solidity
Codebase	https://github.com/NFT-Legend/Legend
Commits	2a85915559ee9519e1545e27d3a97769e5de9497 7a1499ff1759b2d370841ab011d672524a19d38b

Audit Summary

Delivery Date	Jun 08, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

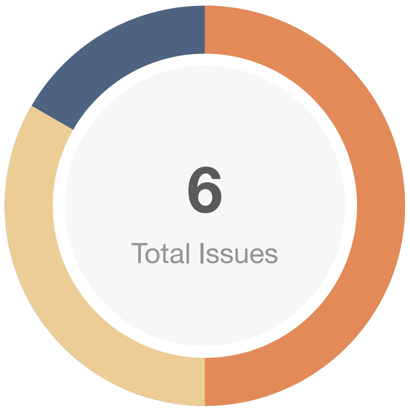
Vulnerability Summary

Total Issues	6
● Critical	0
● Major	3
● Medium	0
● Minor	2
● Informational	1
● Discussion	0

Audit Scope

ID	file	SHA256 Checksum
BPL	BonusPool.sol	82eeb340b5f3bdfbf650e1426a2ab4819805fa0a4cf17c32b1b040d82edbc30a
LGC	LGC.sol	6135a53dcdc72758eed22c57e34ae9ae4ad1b58b6724f8ddc4d873592a852666
SML	StakeMine.sol	6dbdc6fc0ccab9cf0ab606bf37a7b1cb262ee310d781905accb63480b69d3a17

Findings



🟥	**Critical**	**0** (0.00%)
🟧	**Major**	**3** (50.00%)
🟨	**Medium**	**0** (0.00%)
🟨	**Minor**	**2** (33.33%)
🟦	**Informational**	**1** (16.67%)
🟩	**Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
BPL-01	Centralized Risk	**Centralization / Privilege**	● **Major**	**Partially Resolved**
BPL-02	Failed To Adjust Period Awards	Logical Issue	● Minor	✓ Resolved
BPL-03	Centralized Risk	**Centralization / Privilege**	● **Minor**	**Partially Resolved**
LGC-01	Lack of Check for Reentrancy	Logical Issue	● Informational	✓ Resolved
LGC-02	Centralized Risk	**Centralization / Privilege**	● **Major**	**Partially Resolved**
SML-01	Lack of Check for Reentrancy	Logical Issue	● Major	✓ Resolved

BPL-01 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	BonusPool.sol: 131, 137	**Partially Resolved**

Description

The owner of the account that has the `Store` role has the privilege to increase the total value of awards `total` by calling function `injectCapital()` in contract `BonusPool.sol`. Any user who has `Store` role can call `injectCapital()` function to increase `total` without transferring any `token` to this contract and without any awards amount limitation. This `Store` role can be granted by `owner` to any address.

Recommendation

We advise the client to carefully manage the account with `owner` role's private key and avoid any potential risks of being hacked, and make sure the accounts that are granted `Store` role won't do any malicious actions. 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

`[Legend]`: The logic of the game is that the function `injectCapital()` will be invoked when the asset was transferred into the contract, and every user can directly call function `injectCapital()`. There will be a DAO mechanism to manage the access to the project, and the DAO will not be enabled at the initial stage of the project considering the frequent updates of the project.

`[CertiK]`: Based on the update in commit 7a1499ff1759b2d370841ab011d672524a19d38b, user who has `Store` and `PreSale` roles at the same time can still increase the value of `total` without transferring any assets into the contract. As currently DAO will not be enabled, which will allow any user who has `Store` and `PreSale` roles access to the contract. We advise the client to consider release such information and

DAO plan to the community to increase the transparency of the project and also enable DAO as early as possible.

`[Legend]`: DAO module will be enabled once project V3 modules are completed. This plan has been released on the Legend official website https://www.legendnft.com/ and has been notified to the community.

BPL-02 | Failed To Adjust Period Awards

Category	Severity	Location	Status
Logical Issue	● Minor	BonusPool.sol: 85	✓ Resolved

Description

Based on the current design, the total value of the `ratio` must be less than 100 in order to allow the new `awardsInfo.ratio` to be added into the `awardsInfos`. However if the total `ratio` happened to be 100, there's no way to update any `awardsInfo` of array `awardsInfos` as the `require` check `require(awardsInfo.ratio > 0 && awardsInfo.ratio <= 100 - ratio, "Invalid ratio");` in L91 will always fail.

Recommendation

We advise the client to consider adjust the `require` check and allow `awardsInfo.ratio` equal to 0 case.

```
91  require(awardsInfo.ratio <= 100 - ratio, "Invalid ratio");
```

Alleviation

`[Legend]`: `ratio` of a specific award is not allowed to be modified by the original design. Based on suggestions from the legend's community, the implementation of function `addAwards()` has been updated to allow the updates of award info in the suggested design. The update reflects in the commit 7a1499ff1759b2d370841ab011d672524a19d38b

BPL-03 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Minor**	BonusPool.sol: 257, 286, 308	**Partially Resolved**

Description

The owner of the account that has the `Fragment` role has the privilege to decide the value of argument `_from` when calling the function `onERC1155BatchReceived`. The value of `_from` can thus decide the address to where the "Token" and `"Totem"` are minted in function `burnFragment()` and `composite()`. Any user who has `Fragment` role can call `onERC1155BatchReceived()` function to mint those assets to any address. This `Fragment` role can be granted by `owner` to any address.

Recommendation

We advise the client to carefully manage the account with `owner` role's private key and avoid any potential risks of being hacked, and make sure the accounts that are granted `Fragment` role won't do any malicious actions. 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

`[Legend]`: `Fragment` represents the system roles. There will be a DAO mechanism to manage and grant the `Fragment` to a specific group of users, and the DAO will not be enabled at the initial stage of the project considering the frequent updates of the project.

`[CertiK]`: As currently DAO will not be enabled initially, we advise the client to consider release such information and plan to the community to increase the transparency of the project and also enable DAO as early as possible.

`[Legend]`: DAO module will be enabled once project V3 modules are completed. This plan has been released on the Legend official website https://www.legendnft.com/ and has been notified to the community.

LGC-01 | Lack of Check for Reentrancy

Category	Severity	Location	Status
Logical Issue	● Informational	LGC.sol: 121, 134	✓ Resolved

Description

Based on the ERC1363, function `onTransferReceived()` at the token `recipient` address will be called in function `_checkAndCallTransfer()`, and similarly, function `onApprovalReceived()` at the token `spender` address will be called in function `_checkAndCallApprove()`. These may cause reentrancy issue which reentrant into other functions in the contracts. For example, `transferFromAndCall()` will call `transferFrom()` of ERC20, which has `allowance[from][msg.sender] -= amount;` to restrict the maximum amount of token that can be transferred from `from` to `to` by `msg.sender`. However, if `approve[][]` was mistakenly assigned with a huge number which is not expected by the user, the asset of this user may therefore

Recommendation

Although there's `allowance[from][msg.sender] -= amount;` to restrict the maximum amount of token that can be transferred from `from` to `to` by `msg.sender`, we would like to recommend the client to follow the best practice of security and add `nonReentrant` modifier of openzeppelin library to function `_checkAndCallTransfer()` and `_checkAndCallApprove()`, which can protect all other functions that call these two internal functions from reentrancy issue.

Alleviation

`[Legend]`: The client heeded our advice and fixed the issue in commit 7a1499ff1759b2d370841ab011d672524a19d38b

LGC-02 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	LGC.sol: 31, 42, 54	**Partially Resolved**

Description

The account owner of `owner` address has privillege to add specific permits to an arbitrary address in contract `Manager.sol`. Any user who has one type of permit can call `mint()`, `teamClaim()`, or `activityClaim()` function to increase ERC20 asset balance of a specific address `to`, the increased amount is restricted by `remainedSupply`, `team` and `activity` variables though.

Recommendation

We advise the client to carefully manage the account with `owner` role's private key and avoid any potential risks of being hacked. Any sensitive calls to functions `mint()`, `teamClaim()`, and `activityClaim()`. should notify the community with an reasonable window for them to react. 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

`[Legend]`: There will be a DAO mechanism to manage and grant any roles, including `admin`, to any specific user, and the DAO will not be enabled at the initial stage of the project considering the frequent updates of the project.

`[CertiK]`: As currently DAO will not be enabled initially, we advise the client to consider release such information and plan to the community to increase the transparency of the project and also enable DAO as early as possible.

`[Legend]`: DAO module will be enabled once project V3 modules are completed. This plan has been released on the Legend official website https://www.legendnft.com/ and has been notified to the community.

SML-01 | Lack of Check for Reentrancy

Category	Severity	Location	Status
Logical Issue	● Major	StakeMine.sol: 14, 23, 25, 35, 37	✓ Resolved

Description

When an external user calls function `staking(int256)` with a negative value, `recipient.transfer(uint256(-amount));` will transfer the opposite of this negative value amount token to `msg.sender`/`recipient` address, which will thus invoke its `fallback()` or `receive()` function. These two function can be implemented with logic to call `staking(int256)` or `staking(address, int256)` function in `StakeMine` contract again as a reentrant call, which can repeat the process of transferring the opposite of the negative value amount token to `msg.sender`/`recipient` address again. Similarly, `staking(address, int256)` function will also cause the reentrancy issue as the address `poolAddr` is an arbitrary argument and can be of any value, this will bring the unknown logic and results of function `transferFrom()` and `transfer()`. Also for the function `withdraw()`, L35 and L37 may cause the reentrancy issue due to the same reason.

Recommendation

We recommend applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the `staking(int256 amount)` and `staking(address poolAddr, int256 amount)` function to prevent reentrancy attack.

Alleviation

`[Legend]`: The client heeded our advice and fixed the issue in commit 7a1499ff1759b2d370841ab011d672524a19d38b

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.

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

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.