Security Assessment

Liti Capital

Jun 11th, 2021

Table of Contents

Summary

Overview

- Project Summary
- Audit Summary
- Vulnerability Summary
- Audit Scope

Findings

- LIT-01 : Centralization Risks
- LIT-02 : Lack of Return Value Handling
- LIT-03 : Comment Typo
- LIT-04 : Lack of Duplicate Request Check
- LIT-05 : Lack of Input Validation
- LTE-01 : Lack of Return Value Handling

Appendix

Disclaimer

About

Summary

This report has been prepared for Liti Capital 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	Liti Capital
Description	A custom ERC20 smart contracts (LITI, WLITI) with ERC20 Capped Upgradable, Ownable Upgradeable and Pauseable Upgradeable capabilities.
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/Liti-Capital/Equity-Tokenization/tree/main/smart%20contracts/upgradeable/contracts
Commit	530d7296e1d70907e00d0d42f524f094b0ef43ff 36642109aa9de841429db09bfe53d066ce9b445b 7d9d82a0832f4a421dfd0c07bdfdda7281b47373

Audit Summary

Delivery Date	Jun 11, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Total Issues	6
Critical	0
Major	0
Medium	0
Minor	0
Informational	6
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
LIT	LITI.sol	ade588bcbb02816db9cdf57ca7e3e9937597d6f37d9bccf53d9627b776f6cc05
LII	LITIProxy.sol	90cfa73cbb3bb681aacd987a4fa472b6684c1ab2985b13b5567eee58e61faac7
LTE	wLITI.sol	a707d0b5cf85b098fa4e641e74452b6f9077a8069e85e1487b4c9430a9ddef50

There are a few depending injection contracts or addresses in the current project:

`_logic`, `admin_` in the contract `LITIProxy`;

`litiAddress` in the contract `wLITI`.

We assume these contracts or addresses are valid and non-vulnerable actors and implementing proper logic to collaborate with the current project.

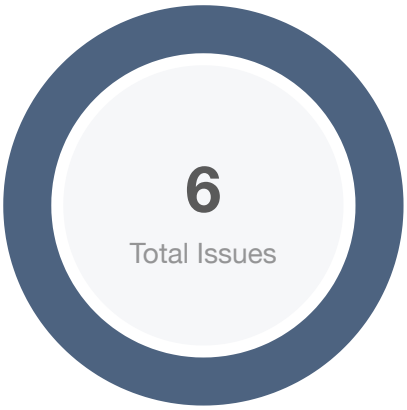
To set up the project correctly, improve overall project quality and preserve upgradability, the following roles are adopted in the codebase:

`owner` is adopted to freeze, unfreeze, approve users, and mint tokens for distribution in the contract `LITI`;

`admin_` is adopted to upgrade the proxy in the contract `LitiProxy`.

To improve the trustworthiness of the project, and dynamic runtime updates in the project should be notified to the community. Any plan to invoke the functions mentioned above should also be considered to move to the execution queue of the `Timelock` contract.

Findings



Critical	0 (0.00%)
Major	0 (0.00%)
Medium	0 (0.00%)
Minor	0 (0.00%)
Informational	6 (100.00%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
LIT-01	Centralization Risks	Logical Issue	Informational	Partially Resolved
LIT-02	Lack of Return Value Handling	Logical Issue	Informational	Resolved
LIT-03	Comment Typo	Coding Style	Informational	Acknowledged
LIT-04	Lack of Duplicate Request Check	Logical Issue, Coding Style	Informational	Acknowledged
LIT-05	Lack of Input Validation	Logical Issue	Informational	Resolved
LTE-01	Lack of Return Value Handling	Logical Issue	Informational	Resolved

LIT-01 | Centralization Risks

Category	Severity	Location	Status
Logical Issue	Informational	LITI.sol: 128, 190, 166, 83~87	Partially Resolved

Description

The ownership of contract `LITI` has been granted with the following authorities:

1. mint new tokens and distribute them to users;
2. hand in `tokenRecoveryRequest` and process the request;
3. freeze users' accounts.

The owner of contract `LITI` is allowed to issue new `LitiCapital` tokens. We understand the project needs ownership to mint tokens and issue share for users. However, the potential attacker can mint an enormous amount of tokens to crash the project ecosystem if he/she obtains the owner's account. In addition, the ownership can hand in `TokenRecoveryRequest` by calling function `tokenRecoveryRequest`, and then call function `recoverToken()` to transfer all tokens from users' accounts to any account. This implementation exposes users' assets to the attacker if ownership is not handled properly. Furthermore, the owner can freeze users' accounts by calling the function `freeze()`. This privilege might discourage users from utilizing their assets in this project.

Recommendation

We advise the client to handle ownership carefully to avoid potential attacks. We also advise the client to consider the following solutions:

1. `Timelock` with reasonable latency for community awareness on privileged operations;
2. DAO or Governance module increasing transparency and community involvement.
3. Multisig with community-voted 3rd-party independent co-signers;

Alleviation

[**Liti Capital Team**]: Given the nature of Liti Capital's business and to address regulatory factors of the company， the admin functions are required in case of requirements made directly to us by authorities. The admin and owner accounts will be implemented as multi-sig wallets, with 3 signers (with the possibility to increase the signers). Initially2 out of 3 signers are required. The signers are the co-founders of Liti Capital and the multi-sig is made to reflect the legal requirement of the company that any administrative

decision must be signed by two of the founders.The wallets of the confounders are hardware wallets and a copy of the private keys of each of the founders will be stored in a safe vault in a Bank in Switzerland.

LIT-02 | Lack of Return Value Handling

Category	Severity	Location	Status
Logical Issue	Informational	LITI.sol: 37	Resolved

Description

Function `approveUser()` is not void-returning function and returns a boolean. However, its returning value has not been handled in function `initialize()`.

Recommendation

We advise the client to handle the return value in function `initialize()` before continuing processing.

Alleviation

The client has resolved the issue by handling the return value of function `approveUser()` with the following `require` statement:

```
    require(approveUser(hex"00", msg.sender));
```

This change is reflected in the commit `36642109aa9de841429db09bfe53d066ce9b445b`.

LIT-03 | Comment Typo

Category	Severity	Location	Status
Coding Style	Informational	LITI.sol: 160, 194	Acknowledged

Description

There are two mis-spelled word in the code comment and error message:

1. Line 160: Mis-spelling word `reassign` as `reasgin`;
2. Line 194: Mis-spelling word `attempt` as `atempt`.

Recommendation

We advise to correct these spelling error.

Alleviation

N/A

LIT-04 | Lack of Duplicate Request Check

Category	Severity	Location	Status
Logical Issue, Coding Style	Informational	LITI.sol: 172~175	Acknowledged

Description

The function `tokenRecoveryRequest()` does not check for duplicate requests stored in `tokenRecoveryRecords`, which means the user can hand in the recovery requests for a specific account multiple times. This might allow the attacker to overwrite the `newAccount` stored and take users' tokens. We understand this risk is not likely to happen because function `tokenRecoveryRequest()` requires ownership to be called. This implementation also allows users to correct their requests. As a result, we will mark this as a discussion for further discussion.

Alleviation

[**Liti Capital Team**]: This implementation has the purpose to allow the recovery of tokens of users that may have lost access to their wallets. The function requires owner access. It is possible to modify the request, for instance, to change the new address to which the tokens will be moved. This is again in case that the original owner requests a change, but this will reinitiate the locking time which is 30 days, which in case of being wrong, will give enough time to the user to contact the company and request respective changes.

LIT-05 | Lack of Input Validation

Category	Severity	Location	Status
Logical Issue	Informational	LITI.sol: 67, 83, 99, 112, 120	Resolved

Description

Before updating the status of the contract (paused or unpaused) or accounts (approved and frozen), validation of the current status should be performed. The following functions do not sanitized the current status before the updates:

- `LitiCapital.approveUser()`
- `LitiCapital.unfreeze()`
- `LitiCapital.freeze()`
- `LitiCapital.PauseContract()`
- `LitiCapital.UnpauseContract()`

Recommendation

We advise the client to check current status before setting a new status when calling the aforementioned functions. For example,

```
67      function approveUser(bytes32 data, address account)
68          public
69          onlyOwner
70          returns (bool)
71      {
72          require(!isApproved[account], "The account has already been approved");
73          isApproved[account] = true;
74          emit UserApproved(account, data);
75          return true;
76      }
```

Alleviation

The client has resolved this issue by adding neccessary `require` statements for input validation. The changes are reflected in commit `36642109aa9de841429db09bfe53d066ce9b445b`.

LTE-01 | Lack of Return Value Handling

Category	Severity	Location	Status
Logical Issue	Informational	wLITI.sol: 29, 43	Resolved

Description

Functions `liti.transferFrom()` and `liti.transfer()` are not void-returning functions. Ignoring its return values, especially when their return values might represent the status if the transaction is executed successfully, might cause unexpected exceptions.

Recommendation

We advise to handle the return values of functions `liti.transferFrom()` and `liti.transfer()` before continuing processing.

Alleviation

The client has resolved the issue by handling the return values with the following `require` statements:

```
29      require(liti.transferFrom(msg.sender, address(this), amount));
```

```
46      require(liti.transfer(msg.sender, amount));
```

These changes are reflected in the commit `7d9d82a0832f4a421dfd0c07bdfdda7281b47373`.

Appendix

Finding Categories

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.

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.