

The logo for DeHacker, featuring a stylized 'D' icon followed by the text 'DeHacker' in a bold, sans-serif font. The 'D' icon is a square with a diagonal line. The text is colored with a green-to-yellow gradient.

**DeHacker**

Security Assessment

**CheersLand**

Mar. 31th, 2022



# Contents

CONTENTS .....	
1 SUMMARY .....	
3 ISSUE CATEGORIES .....	
4 OVERVIEW .....	
	5
PROJECT SUMMARY .....	5
AUDIT SUMMARY .....	5
VULNERABILITY SUMMARY .....	6
AUDIT SCOPE .....	6
UNDERSTANDINGS .....	7
OVERVIEW .....	7
PRIVILEGED FUNCTIONS .....	7
FINDINGS .....	
9 MAJOR .....	
	10
<b>CHEERSLAND-01   CENTRALIZATION RELATED RISKS</b> .....	10
DESCRIPTION .....	10
RECOMMENDATION .....	10
ALLEVIATION .....	11
<b>FIP-01   USERS MAY BE UNABLE TO CLAIM THE TOKENS THEY PURCHASED.</b> .....	12
DESCRIPTION .....	12
RECOMMENDATION .....	12
THE TEAM SHOULD ENSURE USERS CAN GET ALL THE LP ADDRESS THEY PURCHASED. ....	12
ALLEVIATION .....	12
MEDIUM .....	13
<b>FIP-02   POTENTIAL REENTRANCY ATTACK</b> .....	13
DESCRIPTION .....	13
RECOMMENDATION .....	13
ALLEVIATION .....	13
MINOR .....	14
<b>FIP-03   MISSING INPUT VALIDATION</b> .....	14
DESCRIPTION .....	14
RECOMMENDATION .....	14
ALLEVIATION .....	14
<b>FIP-04   UNCHECKED VALUE OF ERC-20 TRANSFER() /TRANSFERFROM() CALL</b> .....	15
DESCRIPTION .....	15
RECOMMENDATION .....	15
ALLEVIATION .....	15
INFORMATIONAL .....	16
<b>FIP-05   FUNCTION VISIBILITY OPTIMIZATION</b> .....	16
DESCRIPTION .....	16
RECOMMENDATION .....	16



ALLEVIATION .....	16
DISCLAIMER .....	17
APPENDIX .....	18
FINDING CATEGORIES .....	18
CHECKSUM CALCULATION METHOD .....	18
ABOUT .....	19



## Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting



## Issue Categories

Every issue in this report was assigned a severity level from the following:

### **Critical severity issues**

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

### **Major severity issues**

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

### **Medium severity issues**

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

### **Minor severity issues**

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

### **Informational**

A vulnerability that has informational character but is not affecting any of the code.



# Overview

## Project Summary

Project Name	CheersLand
Platform	BSC
website	<a href="https://cheersland.org/">https://cheersland.org/</a>
Type	GameFi
Language	Solidity
Codebase	<a href="https://github.com/CheersLand/cheersland-igo-smart-contract">https://github.com/CheersLand/cheersland-igo-smart-contract</a>
Commit	ec525d786518905c37bcf3142901a50d5344abc4

## Audit Summary

Delivery Date	Mar. 31, 2022
Audit Methodology	Static Analysis, Manual Review



## Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	2	0	0	1	0	1
Medium	1	0	0	0	0	1
Minor	2	0	0	0	0	2
Informational	1	0	0	0	0	1
Discussion	0	0	0	0	0	0

## Audit scope

ID	File	SHA256 Checksum
FIP	FundraisingIdoPool.sol	8cf9607fa80dfa8c8de2d2ae2b33220b91c75bec244b13b2ebf0b2e9a4cff4cb



# Understandings

## Overview

CheersLand is a project for users to purchase `lpAddress` token.

Users use `fundraising Address` token to buy the `lpAddress`.

There is a fundraising goal that is set by the team. When the fundraising goal is reached, the team will only receive the designated amount mandated for the fundraising goal. The remaining tokens will be returned to users.

There is a `startTime` and `endTime` which represents the start time and end time of the purchase activity. There is a `claimTime`. After claim time, users can get `lpAddress` they purchased, and reclaim the excess proceeds from the remaining `fundraisingAddress`. The team can only claim the designated amount described per the `fundraising Address` not greater than the fundraising goal and extract the surplus `lpAddress`.

There is a whitelist strategy. Variable `rand` is used to mark users at a whitelist level. `rank` has four values: 0, 1, 2, 3, respectively means: 0: never be a whitelist, 1: normal whitelist, 2: super whitelist, 3: ever a whitelist, and later become not a whitelist. Only users in the whitelist can purchase `lpAddress`. Users in the normal whitelist have a default purchase limit `whitelistQuota`. Users in the super whitelist can have a purchase limit that is set by `operator`.

If the user's mortgage num is greater than the `threshold` in a third pool `poolAddress`, he will be set as a normal whitelist when he purchases `lpAddress`.

Additionally, the contract `FundraisingIdoPool` can work correctly only when there is enough `lpAddress` in it. According to the codebase, the amount of `lpAddress` should be `lpQuantitySold` when the activity starts.

## Privileged Functions

The project contains the following privileged functions. They are used to modify the contract configurations and address attributes. We grouped these functions below:

The `onlyOperator` modifier:

contract `FundraisingIdoPool.sol`:

- function `setPoolAddress()`
- function `setAdminAddress()`
- function `setWhitelistQuota()`
- function `setExchangeRate()`
- function `setUpperLimit()`

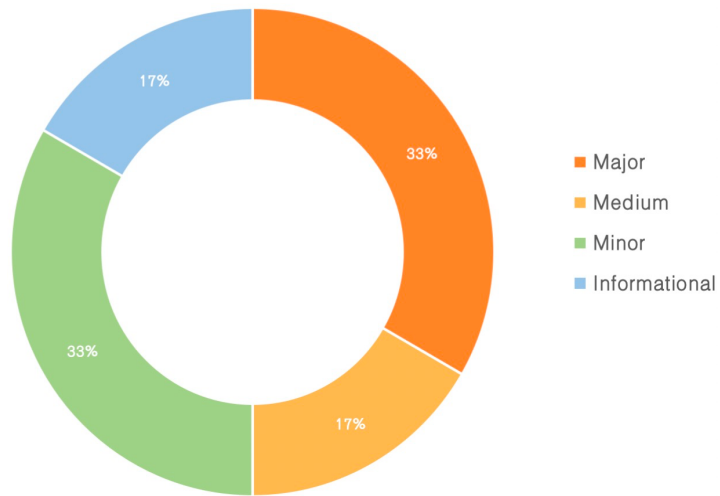




- function setEndTime()
- function setClaimTime()
- function addSuperWhiteList()
- function setWhiteList()
- function ownerClaim()
- function extractSurplusLp()



## Findings



ID	Title	Category	Severity	Status
CheersLand-01	Centralization Related Risks	Centralization / Privilege	Major	Acknowledged
FIP-01	Missing Input Validation	Volatile Code	Minor	Resolved
FIP-02	Function Visibility Optimization	Gas Optimization	Informational	Resolved
FIP-03	Unchecked Value of ERC-20 transfer() / transferFrom() Call	Volatile Code	Minor	Resolved
FIP-04	Potential Reentrancy Attack	Logical Issue	Medium	Resolved
FIP-05	Users May Be Unable To Claim TheTokens They Purchased	Logical Issue	Major	Resolved



# Major

## CheersLand-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	Major	Global	Acknowledged

### Description

In the contract `FundraisingIdoPool`, the role `operator` has the authority over the following function:

1. Update `poolAddress` through `setPoolAddress`.
2. Update `_adminAddress` through `setAdminAddress` function.
3. Update `whiteListQuota` through `setWhiteListQuota` function.
4. Update `exchangeRate` through `setExchangeRate` function.
5. Update `upperLimit[_account]` through `setUpperLimit` function.
6. Update `endTime` through `setEndTime` function.
7. Update `claimTime` through `setClaimTime` function.
8. Set super whitelist through `addSuperWhiteList` function.
9. Set whitelist through `setWhiteList` function.
10. Claim `fundRaisingAddress` through `ownerClaim` function.
11. Extract surplus `lpAddress` through `extractSurplusLp` function.

without obtaining the consensus of the community.

### Recommendation

We advise the client to carefully manage the `owner`, `injector`, `operator` account's private key to 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 smart-contract-based accounts with enhanced security practices, e.g., Multi signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at the different levels in terms of short-term and long-term:

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

## Alleviation

---

No Alleviation



## FIP-01 | Users May Be Unable To Claim The Tokens They Purchased.

---

Category	Severity	Location	Status
Logical Issue	Major	FundraisingPool.sol: <a href="#">271-279</a>	Resolved

### Description

---

Operator can transfer all the `lpAddress` in contract `FundraisingLidoPool` to his own account after `claimTime` through function `extractSurplusLp()`. This operation may be earlier than the users to call `calim()`. This will cause the users unable to get the `lpAddress` they purchased.

### Recommendation

---

The team should ensure users can get all the `lpAddress` they purchased.

### Alleviation

---

The team heeded our advice and removed the `extractSurplusLp` function. Added a logic in function `ownerClaim` to ensure there will be enough `lpAddress` in the contract for users to claim. Code change was applied in commit: `d333c208cc2f1c9f52244f66773eda6f9fd7d3ad`.



## Medium

### FIP-02 | Potential Reentrancy Attack

Category	Severity	Location	Status
Logica Issue	Medium	FundraisingIdoPool.Sol: <a href="#">239-246. 262-266</a>	Resolved

### Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

### Recommendation

We recommend using the [Checks-Effects-Interactions Pattern](#) to avoid the risk of calling unknown contracts or applying OpenZeppelin [ReentrancyGuard](#) library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

### Alleviation

The team heeded our advice and added an inspector lock. Code change was applied in commit: d333c208cc2f1c9f52244f66773eda6f9fd7d3ad.



## Minor

### FIP-03 | Missing Input Validation

---

Category	Severity	Location	Status
Volatile Code	Minor	FundraisingIdoPool.Sol: <a href="#">92.93.99.108.112</a>	Resolved

### Description

---

The given input is missing the check for the non-zero address.

### Recommendation

---

We advise adding the check for the passed-in values to prevent unexpected error as below:

### Alleviation

---

The team heeded our advice and added a zero check. Code change was applied in commit: `d333c208cc2f1c9f52244f66773eda6f9fd7d3ad`.



## FIP-04 | Unchecked Value of ERC-20 `transfer()` / `transferFrom()` Call

---

Category	Severity	Location	Status
Volatile Code	Minor	FundraisingIdoPool.Sol: <a href="#">213.240.243.263.276</a>	Resolved

### Description

---

The linked `transfer()` / `transferFrom()` invocations do not check the return value of the function call which should yield a true result in case of proper ERC-20 implementation.

### Recommendation

---

As many tokens do not follow the ERC-20 standard faithfully, they may not return a bool variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that [OpenZeppelin's SafeERC20.sol](#) implementation is utilized for interacting with the `transfer()` and `transferFrom()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

### Alleviation

---

The team heeded our advice and change to `safeTransfer` and `safeTransferFrom`. Code change was applied in commit: `d333c208cc2f1c9f52244f66773eda6f9fd7d3ad`.





# Informational

## FIP-05 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	FundraisingIdoPool.Sol: <a href="#">136.</a>	Resolved

### Description

The following functions are declared as `public`, contain array function arguments, and are not invoked in any of the contracts contained within the project's scope. The functions that are never called internally within the contract should have external visibility.

### Recommendation

We advise that the functions' visibility specifiers are set to `external`, and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

### Alleviation

The team heeded our advice and change `public` to `external`. Code change was applied in commit: `d333c208cc2f1c9f52244f66773eda6f9fd7d3ad`.



## Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on thereports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content , and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.



# 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.

### Coding Style

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

### Volatile Code

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

### 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.



## About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

### BLOCKCHAINS



Ethereum



Cosmos



Eos



Substrate

### TECH STACK



Python



Rust



Solidity



C++

### CONTACTS

<https://dehacker.io><https://twitter.com/dehackerio>[https://github.com/dehacker/audits\\_public](https://github.com/dehacker/audits_public)<https://t.me/dehackerio><https://blog.dehacker.io/>



**DeHacker**

March 2022