DeHacker

Code Security Assessment

LIFE Crypto Deploy

March25th, 2023

Contents

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	LIFE Crypto Deploy
Platform	BSC
website	https://lifedefi.co/
Type	DeFi
Language	Solidity

Vulnerability Summary

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

Audit scope

ID	File	SHA256 Checksum
LCI	LifeCrypto.sol	6bd0e6ffcc1e805aa10ba967a65b5536263605f061de5388f3a255078d282ff3

Findings

ID	Category	Severity	Status
LCI-01	Centralization /Privilege	Major	Acknowledged
LCI-02	Logical Issue	Medium	Resolved

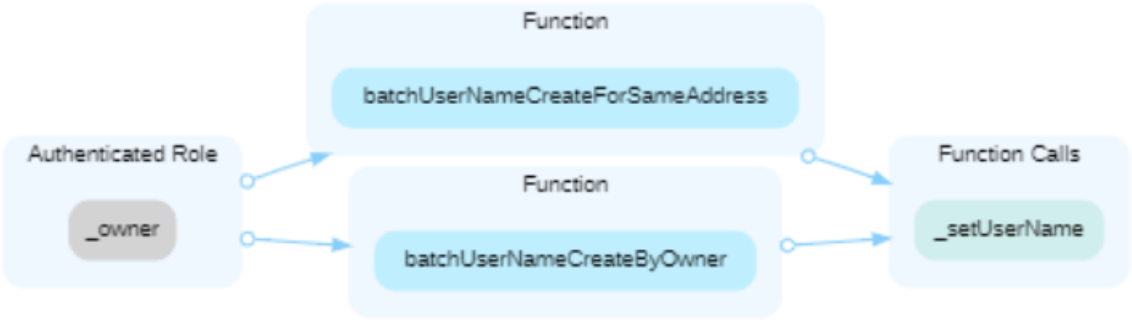
Major

LCI-01 | CENTRALIZATION RISKS IN LIFECRYPTO.SOL

Category	Severity	Location	Status
Centralization / Privilege	Major	LifeCrypto.sol	Acknowledged

Description

In the contract LIFECRYPTO the role _owner has authority over the functions shown in the diagram below.



AND the role CREATE_CUSERNAME_ROLE has authority over the following function(s):
function createCustomUsername() , to create a username for a wallet.
AND the role CREATE_PUSERNAME_ROLE has authority over the following function(s):function createPaidUsername() , to create a username and record its payment information.
AND the role USERNAME_TRANSFER_FREEZE_ROLE has authority over the following function(s):
function freezeUserName() and unFreezeUserName() , to freeze or unfreeze the username.
AND the role BIND_USER_PAYMENT_ROLE has authority over the following function(s):
function bindUserPayment() , to binding one payment information and one username.
AND the role REMOVE_PAYMENT_ROLE has authority over the following function(s):
function removeUserPayment() , to remove the binding of username and transaction info.
AND the role DEFAULT_ADMIN_ROLE has authority over the following function(s):
function createCustomUsername() , to create a username for a wallet.
function createPaidUsername() , to create a username and record its payment information

function freezeUserName() and unFreezeUserName() , to freeze or unfreeze the username.
function transferUserName() , to transfer a username.
function bindUserPayment() , to binding one payment information and one username.
function removeUserPayment() , to remove the binding of username and transaction info.
function updateUserInfo() , to change the full name of the user.
function grantRole() and revokeRole() , to grant roles to or revoke roles from any accounts.

Any compromise to the privileged accounts may allow the hacker to take advantage of this authority and bring unpredictabledamages to the project.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level ofdecentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefullymanage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommendcentralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accountswith enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that wouldalso mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:
Timelock and Multi sign (2/3, 3/5) combination mitigate by delaying the sensitive operation and avoiding a single point of keymanagement failure.
Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND
Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private keycompromised;
AND
A medium/blog link for sharing the timelock contract and multi-signers addresses information with the publicaudience.

Long Term:
Timelock and DAO, the combination, mitigate by applying decentralization and transparency.
Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND
Introduction of a DAO/governance/voting module to increase transparency and user involvement.
AND
A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:
Renouncing the ownership or removing the function can be considered fully resolved.
Renounce the ownership and never claim back the privileged roles.
OR
Remove the risky functionality.

Medium

LCI-02 LOGICAL ISSUE OF FUNCTION createRandomUsername()

Category	Severity	Location	Status
Logical Issue	Medium	LifeCrypto.sol: 28	Resolved

Description

The function createRandomUsername() is not really a function to offer random names. It only creates names from user1 touserN. And if the target user name is taken by a custom username. This function will always revert.

Recommendation

We recommend the team check the logic and fix the issue.

Optimizations

ID	Category	Severity	Status
LCI-03	Gas Optimization	Optimization	Resolved

Optimization

LCI-03 | INEFFICIENT MEMORY PARAMETER

Category	Severity	Location	Status
Gas Optimization	Optimization	LifeCrypto.sol: 35, 41, 41, 50, 50, 60, 67, 73, 79, 93, 93, 100,108, 173, 187, 196, 202, 208	Resolved

Description

One or more parameters with memory data location are never modified in their functions and those functions are nevercalled internally within the contract. Thus, their data location can be changed to calldata to avoid gas consumptioncopying from calldata to memory.

```
35      function createCustomUsername(string memory userName, address walletAddress)
public returns (bool) {
```

`createCustomUsername` has memory location parameters: `userName`.

```
41      function createPaidUsername(string memory txid,string memory userName,
address walletAddress) public returns (bool) {
```

`createPaidUsername` has memory location parameters: `txid`, `userName`.

```
50      function batchUserNameCreateByOwner(string[] memory userNames, address[]
memory walletAddresses) public onlyOwner returns (bool) {
```

`batchUserNameCreateByOwner` has memory location parameters: `userNames`, `walletAddresses`.

```
60      function batchUserNameCreateForSameAddress(string[] memory userNames,
address walletAddresses) public onlyOwner returns (bool) {
```

`batchUserNameCreateForSameAddress` has memory location parameters: `userNames`.

```
67      function freezeUserName(string memory userName) public returns (bool) {
```

`freezeUserName` has memory location parameters: `userName`.

```
73      function unFreezeUserName(string memory userName) public returns (bool) {
```

`unFreezeUserName` has memory location parameters: `userName` .

```
 79      function transferUserName(string memory userName, address fromAddress,
address toAddress) public returns (bool) {
```

`transferUserName` has memory location parameters: `userName` .

```
 93      function bindUserPayment(string memory txid, string memory username) public
returns (bool) {
```

`bindUserPayment` has memory location parameters: `txid` , `username` .

```
100      function removeUserPayment(string memory txid) public returns (bool) {
```

`removeUserPayment` has memory location parameters: `txid` .

```
108      function updateUserInfo( address addressVal,string memory fullName) public
returns (bool) {
```

`updateUserInfo` has memory location parameters: `fullName` .

```
173      function checkUserNameAndAddress(string memory userName,address
walletAddress) public view virtual returns (bool){
```

`checkUserNameAndAddress` has memory location parameters: `userName` .

```
187      function checkExistsUserName(string memory userName) public view virtual
returns (bool){
```

`checkExistsUserName` has memory location parameters: `userName` .

```
196      function addressByUserName(string memory username) public view virtual
returns(address){
```

`addressByUserName` has memory location parameters: `username` .

```
202      function isUserNameFreeze(string memory username) public view virtual
returns (bool) {
```

`isUserNameFreeze` has memory location parameters: `username` .

```
208     function getUsernameByPayment(string memory txid) public view virtual
returns (string memory) {
```

`getUsernameByPayment` has memory location parameters: `txid` .

Recommendation

We recommend changing the parameter's data location to calldata to save gas.

For Solidity versions prior to 0.6.9, since public functions are not allowed to have calldata parameters, the functionvisibility also needs to be changed to external .

For Solidity versions prior to 0.5.0, since parameter data location is implicit, changing the function visibility toexternal will change the parameter's data location to calldata as well.

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 the reports 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.

BLOCKCHAIINS

Ethereum
Cosmos
Eos
Substrate

TECH STACK

Python
Solidity
Rust
c++

CONTACTS

https://dehacker.io
https://twitter.com/dehackerio
https://github.com/dehacker/audits_public
https://t.me/dehackerio
https://blog.dehacker.io/

DeHacker

March 2023