DeHacker

Code Security Assessment

Label Foundation

Aug 2 8 th, 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	Label Foundation
Platform	bsc
Website	https://label.foundation/
Type	NFT
Language	Solidity

Vulnerability Summary

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

Audit scope

ID	File	SHA256 Checksum
BEP	BEP20Label.sol	77430a3aac75d47b253f01e4443b4bb391e19170b5659776daf0ce1fa704996b

Findings

ID	Category	Severity	Status
BEP-01	Centralization /Privilege	Medium	Acknowledged
BEP-02	Coding Style	Informational	Acknowledged
BEP-03	Coding Style	Informational	Acknowledged

MEDIUM

CCK-01 | Missing Emit Events

Category	Severity	Location	Status
Centralization / Privilege	Medium	LabelFoundation/ BEP20Label.sol (eaa5801): 364	Acknowledged

Description

In the contract BEP20Label , the deployer has the authority over the following function.constructor() : mint total tokens to the owner

Recommendation

We advise the client to carefully manage the privileged account's private key to avoid any potential risks ofbeing hacked.

In general, we strongly recommend centralized privileges or roles in the protocol to be improved via adecentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g.,Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the differentlevel in term 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 theprivate key;

Introduction of a DAO/governance/voting module to increase transparency and user involvement.

INFORMATIONAL

BEP-02 | Proper usage of “ public ” and “ external ” type

Category	Severity	Location	Status
Coding Style	Informational	LabelFoundation/BEP20Label.sol (eaa5801)	Acknowledged

Description

“ public ” functions that are never called by the contract should be declared “ external ” . When the inputs arearrays, “ external ” functions are more efficient than “ public ” functions.
Examples:
Functions like :
contract BEP20Label : renounceOwnership() , transferOwnership() , increaseAllowance () ,decreaseAllowance()

Recommendation

We recommend using the “ external ” attribute for functions never called from the contract.

INFORMATIONAL

BEP-03 | Too many digits

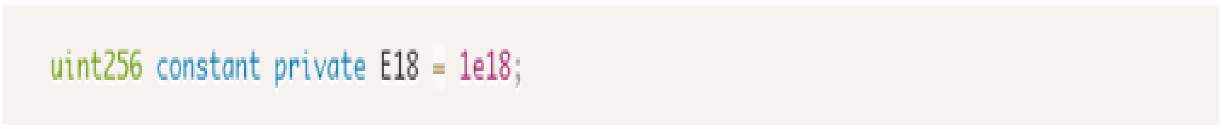
Category	Severity	Location	Status
Coding Style	Informational	LabelFoundation/ BEP20Label.sol (eaa5801): 348, 363	Acknowledged

Description

Literals with many digits are difficult to read and review.

Recommendation

We recommend modifying as below example:



```
uint256 constant private E18 = 1e18;
```

Disclaimer

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

Aug 2023