BLOX STAKING Audit: Vesting and DEX Contracts

July 20, 2021 Heloisa Ceni No Comments

Reading Time: 2 minutes

Contents

1. Introduction
1.1. Summary
1.2. Contracts
1.3. Severity Classification
2. Issues Found by Severity
2.1. Critical severity
2.2. Medium severity
2.3. Minor severity
2.4. Observations
2.5. Conclusion

Introduction

CoinFabrik was asked to audit the contracts for the Vesting for Blox Staking. First we will provide a summary of our discoveries and then we will show the details of our findings.

Summary

The contracts audited are from the Github repository at https://github.com/bloxapp/ssv-network. The audit is based on the commit 95a79e44bbeb35b8c6a2bbd20e47762a4bb0dd11, fixes were applied and the commit ca99df216dc22d1e2170067b714c79d20aa003e0 was used to validate changes.

Contracts

The audited contracts are:

- Misuse of the different call methods
- Integer overflow errors
- Division by zero errors
- Outdated version of Solidity compiler
- Front running attacks
- Reentrancy attacks
- Misuse of block timestamps
- Softlock denial of service attacks
- Functions with excessive gas cost
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Failure to use a withdrawal pattern
- Insufficient validation of the input parameters
- Incorrect handling of cryptographic signatures

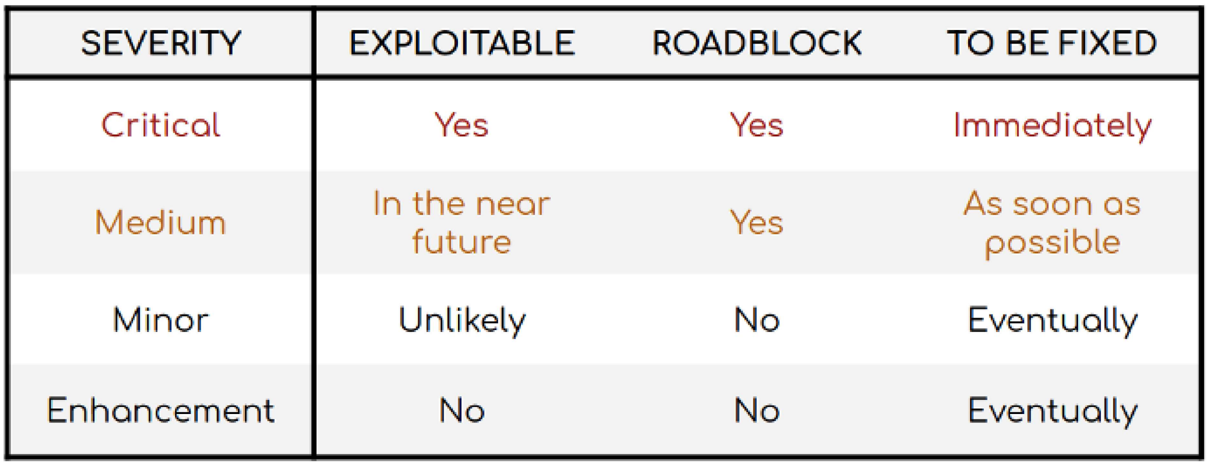
Severity Classification

Security risks are classified as follows:

- **Critical:** These are issues that we manage to exploit. They compromise the system seriously. They must be fixed immediately.
- **Medium:** These are potentially exploitable issues. Even though we did not manage to exploit them or their impact is not clear, they might represent a security risk in the near future. We suggest fixing them as soon as possible.
- **Minor:** These issues represent problems that are relatively small or difficult to take advantage of but can be exploited in combination with other issues. These kinds of issues do not block deployments in production environments. They should be taken into account and be fixed when possible.

● **Enhancement:** These kinds of findings do not represent a security risk. They are best practices that we suggest to implement.

This classification is summarized in the following table:



SEVERITY	EXPLOITABLE	ROADBLOCK	TO BE FIXED
Critical	Yes	Yes	Immediately
Medium	In the near future	Yes	As soon as possible
Minor	Unlikely	No	Eventually
Enhancement	No	No	Eventually

Issues Found by Severity

Critical severity

CR-01 Denial Of Service in `vestings[]` struct

A malicious user could create an arbitrary amount of 'token vestings' for a specific user, all vesting 0 amount, thus growing the size of the vestings struct so that calls to any function looping through it would take a large amount of gas. As a result, these functions will fail. Examples include *`totalVestingBalanceOf(),`* *`revokeAll()`* and *`withdrawFor().`*

Recommendation

Creating a vesting should require a positive amount, or even an amount bigger than a contract-defined threshold.

Status

The recommendation was followed and the issue has been fixed.

Medium severity

No issues found in this category.

Minor severity

MI-01 DEX contract allowing 0 rate

Failure to initialize DEX rate in 0 could lead to overflows and unexpected results.

Recommendation

Add a ‘require’ statement checking that the rate is nonzero.

Status

The recommendation was followed and the issue has been fixed.

Observations

The SSV token is upgradeable (see, for example, link) and therefore its contract could be changed by the owner. We advise care with this, as the expected behaviour may change with any upgrade

Conclusion

We found the contracts to be simple, straightforward and with adequate documentation. A critical vulnerability was found allowing malicious users to impair the availability of certain functions, including withdrawFor(), a fix was proposed and applied. A minor-severity issue was found in the initialization of a token which was also fixed. No issues remain.

Disclaimer: This audit report is not a security warranty, investment advice, or an approval of the BLOX STAKING project since CoinFabrik has not reviewed its platform. Moreover, it does not provide a smart contract code faultlessness guarantee.

Related Posts

RCN Smart Contracts Audit v2

The smart contracts that have been audited were taken from the RCN repository at: https://github.com/ripio/rcn-network/tree/v2....

DreamTeam Smart Contracts Audit for Players' Compensation

Coinfabrik was asked to audit the smart contracts for the DreamTeam system. The DreamTeam contracts...

Cryptosolartech Security Audit

Coinfabrik's smart contract audit's team was asked to audit the contracts for the Cryptosolartech sale....

EtherParty Smart Contract Security Audit

Coinfabrik team has been hired to audit Etherparty smart contracts. Firstly, we will provide a...

Categorized in: Auditoria Smart Contracts, Smart Contract Audit, Smart Contract Audit, Smart Contracts

ALSO ON **COINFABRIK**

What I Have Learned From My Arbitrage …

3 years ago • 2 comments

Lately, lots of posts have been talking about how traders get rich from …

Smart Contract Audits: The Ultimate …

3 years ago • 1 comment

Smart contract audits are needed for the token to be approved by the …

1Inch Exchange Smart Contract Audit

10 months ago • 1 comment

CoinFabrik was asked to audit the contracts for the 1Inch Exchange. First we …

Au

2 ye

Afte
Co
vs.

1 Comment CoinFabrik Disqus' Privacy Policy 1 Login

Recommend Tweet Share Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Andrew Gohan • a month ago

Guys, check out Gain Protocol. Solid project across the board with a small but a good team at the helm

1 | • Reply • Share ›