SOLIDBLOCK

SolidBlock Security Analysis

by Pessimistic

This report is public.

Published: August 19, 2021

Abstract

In this report, we consider the security of smart contracts of SolidBlock project. Our task is to find and describe security issues in the smart contracts of the platform.

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, security audit is not an investment advice.

Summary

In this report, we considered the security of SolidBlock smart contracts. We performed our audit according to the procedure described below.

The initial audit showed an Overpowered role issue that makes possible to censor transactions for certain accounts, and several issues of low severity.

After the initial audit, the code base was updated. Most of the issues were fixed. Also, the documentation for the project was updated.

General recommendations

We recommend adding CI to run tests, calculate code coverage, and analyze code with linters and security tools.

Project overview

Project description

For the audit, we were provided with SolidBlock project on a private GitHub repository, commit 445dc5f6c20ca32e0861b7a69a12d1e03225f5c0.

The documentation for the project was provided as a separate file:
Private Token SW design ver6.0.pdf,
sha1sum is ce12d225bb7dd58d4ab1c7fab870e306cc984852.

The project compiles without any issues. However, optimization is not enabled in **hardhat.config.js** file.

All tests pass, the code coverage is close to 100%.

The total LOC of audited sources is 334.

Updated code base

After the initial audit, the code base was updated. For the recheck, we were provided with commit bdd7f8f1f62484b30206dd3cb6b7bf4ff13a678b.

Also, the developers provided an updated documentation:
Private Token SW design ver6.2.pdf,
sha1sum is 2538dd59a82959de244487a1ed722358e5a1a48a.

Procedure

In our audit, we consider the following crucial features of the code:

1. Whether the code is secure.
2. Whether the code corresponds to the documentation (including whitepaper).
3. Whether the code meets best practices.

We perform our audit according to the following procedure:

- Manual audit
  - We manually analyze code base for security vulnerabilities.
  - We assess overall project structure and quality.
- Report
  - We reflect all the gathered information in the report.

Manual analysis

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger smart contracts security. We highly recommend fixing them.

The audit showed no critical issues.

Medium severity issues

Medium issues can influence project operation in current implementation. We highly recommend addressing them.

Overpowered role

In **openzeppelin-v2.5.1/access/roles/WhitelistedRole.sol**, a user with `WhitelistAdmin` role can add accounts to or remove from the whitelist. Therefore, this role allows to censor transfers from or to the accounts that were whitelisted.

In the current implementation, the system depends heavily on the owner of the contract. Thus, there are scenarios that may lead to undesirable consequences for investors, e.g. if the owner's private keys become compromised.

We recommend designing contracts in a trustless manner or implementing proper key management, e.g. multisig.

> *Comment from developers: In order to address the issue mentioned above, we will use a multi-signature solution. Using Gnosis Safe wallet.*
>
> *Each wallet uses different service providers, separated from any other assets or any other kind of tokens.*
>
> *The legal structure of the minimum required signatures and the entities holding them is currently being defined by our legal team and will be made public with the following updates.*

Low severity issues

Low severity issues can influence project operation in future versions of code. We recommend taking them into account.

Code quality (fixed)

- Consider declaring `public` functions as `external` where possible.
- In **ERC1404Messages** contract, `_codes` field of `Message` struct is unused. Consider removing it.
- NatSpecs are incomplete.

The issues have been fixed and are not present in the latest version of the code.

Project management

- Optimization is not enabled in **hardhat.config.js**.

  The issue has been fixed and is not present in the latest version of the code.
- Two versions of the same OpenZeppelin library are copied into the repository. Consider refactoring the contracts so that they require only one version, which can be provided as npm dependency.

This analysis was performed by Pessimistic:

Evgeny Marchenko, Senior Security Engineer
Igor Sobolev, Security Engineer
Boris Nikashin, Analyst
Irina Vikhareva, Project Manager

August 19, 2021