VDOC Smart Contracts Audit

Audit made by: Merunas Grincalaitis merunasgrincalaitis@gmail.com

Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The contracts audited are the ones available on the github: https://github.com/BlueBikeSolutions/ico at the version specified by the pull request: https://github.com/BlueBikeSolutions/ico/pull/2

Overview

VDOC is the token used by dutyof.care which is a blockchain solution for protecting individuals and helping people who are at risk of being abused, such as elders and people with disabilities. Because the tools provided by the government to protect people in those situations are not working as they should. They are increasing the compliance expectations which creates confusion to businesses.

The solution that the team proposes is to create a dapp that helps businesses to comply with the new regulations when it comes to protecting people. To do so, they have created an ERC20 token called VDOC, which will be used to incentivize and reward behaviors inside the platform to increase the usage and expand the user base of dutyof.care.

There are 44 Smart Contracts for the token and ICO. However most of them come from a fork made to the well-audited contracts of TokenMarketNet. This audit covers the 4 edited Smart Contracts from that fork that have been modified to adapt to this particular crowdsale. These are:

- **Crowdsale.sol**: The main token sale contract for selling the tokens in exchange of ether.
- **CrowdsaleBase.sol**: The main code that will be used in the crowdsale.
- **PricingStrategy.sol**: A contract with several utilities to check if a user is purchasing in the presale or later.
- **TokenTranchePricing**: A contract to define tranches with different prices for the ICO.

At the time of publishing this audit, most of the suggestions have been taken care of and now the code is cleaner and more secure.

Nice Features

The code is pretty simple and straight forward, which is essential for guaranteeing the security of the funds raised. The more complexity, the bigger the risk of making a mistake.

General Improvements

The code in the Smart Contracts is designed for an outdated version of solidity because it's using statements such as throw and doesn't specify the version of the compiler used at the top of each file. This is potentially dangerous since we are dealing with possible bugs from older versions. It also makes more difficult to maintain the code when you want to add new features because you have to deal with an outdated way of coding.

Analysis of the Smart Contracts

In order to check for the security of the contract, we tested several attacks to make sure that the contract is secure and follows

best practices. Here's what we found.

Critical issues found in the Smart Contracts

No critical issues found.

Medium issues found in the Smart Contracts

No medium severity issues found.

Low severity issues found in the Smart Contracts

A lack of require checks in some functions. Nothing serious.

Line by line comments

These are the errors, problems and interesting ideas found in all the Smart Contracts.

Crowdsale.sol

The only addition to this contract is the fallback function that allow users to purchase tokens directly by sending ether to the contract. The rest of the code is from the forked ICO code, which was audited several times.
Line 143: It'd be great to specify the visibility of the fallback function to avoid warnings when compiling the code and to help people understand who should use this function. Since right now, it doesn't have any explicit visibility.
In general the code is good but it's not as consistent as it should be. Some functions are well-documented others aren't.

CrowdsaleBase.sol

Line 182: You're not checking if the arguments of the function are set properly. Even though it's not necessary in this case because you can whitelist any address, it'd nice to check if the address is not empty with something like this: `require(_address != 0x0);`
Line 189: Same situation as before.
Line 197 and 204: Again, it's recommended to check if the arguments of the functions are empty or not.
Line 249: This is a nice implementation to limit the amount of tokens a user can have in total.

PricingStrategy.sol
Nothing to add. There's only a new function and it look great.

TokenTranchePricing.sol
Line 132: The documentation in that line is incomplete, it lacks the description of the return value.

Summary of the audit

They are working with an already tested and audited base of code which really helps to understand the changes made by the team. They made the necessary adjustments without adding too much complexity, which is nice. In general, a very good implementation of a crowdsale with some minor issues that can be easily fixed before deploying the code. My recommendation is to update all the code to the latest version of solidity for maintaining the code in the future.