CERTIK

# CertiK Final Audit Report for VNDC

## Contents

## Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Verification Services Agreement between CertiK and VNDC(the “Company”), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, the “Agreement”). This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK’s prior written consent.

## About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, CertiK’s mission of every audit is to apply different approaches and detection methods, ranging from manual, static, and dynamic analysis, to ensure that projects are checked against known attacks and potential vulnerabilities. CertiK leverages a team of seasoned engineers and security auditors to apply testing methodologies and assessments to each project, in turn creating a more secure and robust software system.

CertiK has served more than 100 clients with high quality auditing and consulting services, ranging from stablecoins such as Binance’s BGBP and Paxos Gold to decentralized oracles such as Band Protocol and Tellor. CertiK customizes its engineering tool kits, while applying cutting-edge research on smart contracts, for each client on its project to offer a high quality deliverable.  For more information: https://certik.io.

## Executive Summary

This report has been prepared for **VNDC** to discover issues and vulnerabilities in the source code of their **ERC-20 Smart Contracts** as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic Analysis, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

## Testing Summary

SECURITY LEVEL

Smart Contract Audit
This report has been prepared as a product of the Smart Contract Audit request by VNDC.
This audit was conducted to discover issues and vulnerabilities in the source code of the VNDC's ERC-20 Smart Contracts.

| | |
|---|---|
| TYPE | Smart Contract |
| SOURCE CODE | [https://github.com/stably-vndc/vndc-token/](https://github.com/stably-vndc/vndc-token/) |
| PLATFORM | EVM |
| LANGUAGE | Solidity |
| REQUEST DATE | Jun 14, 2020 |
| DELIVERY DATE | Aug 23, 2020 |
| METHODS | A comprehensive examination has been performed using Dynamic Analysis, Static Analysis, and Manual Review. |

## Review Notes

### Introduction

CertiK team was contracted by the VNDC team to audit the design and implementation of their token smart contracts and its compliance with the EIPs it is meant to implement.

The audited source code link is:

- https://github.com/stably-vndc/vndc-token/tree/certik-audit/token-contracts/contracts

The goal of this audit was to review the Solidity implementation for its business model, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

### Documentation

The sources of truth regarding the operation of the contracts in scope were minimal although the token fulfilled a simple use case we were able to fully assimilate. To help aid our understanding of each contract's functionality we referred to the thorough in-line comments and naming conventions.

## Summary

The codebase of the project is a typical EIP20 implementation with additional support for pausing and issuance control mechanisms.

Although **certain optimization steps** that we pinpointed in the source code mostly referred to coding standards and inefficiencies, **any minor (or above) flaws** that were identified, **were remediated as soon as possible to ensure the security of the contracts** of VNDC’s team.

The codebase of the project strictly adheres to the standards and interfaces imposed by the OpenZeppelin open-source libraries and as such its typical ERC-20 functions **can be deemed to be of high security and quality, however, the custom functionality built on top of it possessed flaws** we identified.

VNDC’s team considered our references and opted to change their codebase to alleviate these problems. Although one exhibit was not remediated, **we approve of this stance**, as VNDC’s team showed awareness and maturity towards the specific problem.

## Recommendations

Overall, the codebase of the contracts should be refactored to assimilate an additional specification, that of the subtoken tracking mechanism.

## Findings

### Exhibit 1

| TITLE | TYPE | SEVERITY | LOCATION |
|---|---|---|---|
| Visibility Specifier(s) Missing | Language Specific Issue | Informational | VNDCImplementation.sol Lines 44, 45, 49, 50, 53, 54 |

**[INFORMATIONAL] Description:**

The linked variable(s) are contract-level definitions yet lack a visibility specifier.

**Recommendations:**

We advise that a strict visibility specifier is set for them to aid in the legibility of the codebase.

## Exhibit 2

| TITLE | TYPE | SEVERITY | LOCATION |
|---|---|---|---|
| Unused Event Declarations | Optimization | Informational | VNDCImplementation.sol Lines 116, 117 |

**[INFORMATIONAL] Description:**

The aforementioned lines contain event declarations that have not been utilized across the repository.

**Recommendations:**

We suggest that they are properly emitted in the corresponding *“addLabel”* and *“removeLabel”* functions.

## Exhibit 3

| TITLE | TYPE | SEVERITY | LOCATION |
|---|---|---|---|
| Illogical Conditional | Coding Style | Informational | VNDCImplementation.sol Line 443 |

**[INFORMATIONAL] Description:**

The conditional of the “while” loop will always evaluate to true as the variable remainder is a “uint256” , meaning it is capable of representing non-negative numbers.

**Recommendations:**

We advise the team to consider changing the conditional of the “while” loop

## Exhibit 4

| TITLE | TYPE | SEVERITY | LOCATION |
|---|---|---|---|
| Incorrect “require” Check | Optimization | Informational | VNDCIssuer.sol<br>Line 137 |

**[INFORMATIONAL] Description:**

TThe current require check permits a total of _maxMembers + 1 members to be present within the contract, exceeding the number represented by _maxMembers.

**Recommendations:**

We advise that the logical comparison is instead changed to a less-than comparison ( < ).

## Exhibit 5

| TITLE | TYPE | SEVERITY | LOCATION |
|---|---|---|---|
| Unlocked Compiler Version | Language Specific Issue | Informational | VNDCProxy.sol Line 1 |

**[INFORMATIONAL] Description:**

An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers.

This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

**Recommendations:**

We advise that the compiler version is instead locked at the lowest version possible that the full project can be compiled at.

## Exhibit 6

| TITLE | TYPE | SEVERITY | LOCATION |
|---|---|---|---|
| Comment Ambiguity | Coding Style | Informational | VNDCSubtoken.sol Line 40 |

**[INFORMATIONAL] Description:**

Ethereum blockchain explorers and subsequently token trackers utilize the ERC20 Transfer event to track any transmission of funds and accurately depict the total balance of an address. In case that "labelled" tokens are meant to exist independently and would preferably be tracked by explorers autonomously, a Transfer event indeed needs to be emitted at this point. This would also require further changes in the codebase of VNDCImplementation.sol to make sure that explorers properly track the tokens of each label.

**Recommendations:**

We advise that you relay to us exactly how you envision labelled subtokens to behave so we can properly assess whether and what further changes would be necessary.