# Verasity ERC-777 Token Smart Contract Audit

**27 January 2021**

Public

iosiro.com

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to: (i) cybersecurity vulnerabilities and issues in the smart contract source code analysed, the details of which are set out in this report, (Source Code); and (ii) the Source Code compiling, deploying and performing the intended functions. In order to get a full view of our findings and the scope of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

**DISCLAIMER:** By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Zenoic Proprietary Limited trading as "Iosiro" and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Iosiro) owe no duty of care towards you or any other person, nor does Iosiro make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Iosiro hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for

purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Iosiro hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Iosiro, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

# 1. Introduction

iosiro was commissioned by Verasity to conduct a smart contract audit on their VERA ERC-777 token smart contract. The audit was performed on 20 and 27 January 2021.

This report is organized into the following sections.

- **Section 2 - Executive summary:** A high-level description of the findings of the audit.
- **Section 3 - Audit details:** A description of the scope and methodology of the audit.
- **Section 4 - Design specification:** An outline of the intended functionality of the smart contracts.
- **Section 5 - Detailed findings:** Detailed descriptions of the findings of the audit.

The information in this report should be used to better understand the risk exposure of the smart contracts, and as a guide to improving the security posture of the smart contracts by remediating issues identified. The results of this audit are only a reflection of the source code reviewed at the time of the audit and of the source code that was determined to be in-scope.

The purpose of this audit was to achieve the following:

- Identify potential security flaws.
- Ensure that the smart contracts functioned according to the documentation provided.

Assessing the off-chain functionality associated with the contracts, for example, backend web application code, was out of scope of this audit.

Due to the unregulated nature and ease of transfer of cryptocurrencies, operations that store or interact with these assets are considered very high risk with regards to cyber attacks. As such, the highest level of security should be observed when interacting with these assets. This requires a forward-thinking approach, which takes into account the new and experimental nature of blockchain technologies. Strategies that should be used to encourage secure code development include:

- Security should be integrated into the development lifecycle and the level of perceived security should not be limited to a single code audit.
- Defensive programming should be employed to account for unforeseen circumstances.
- Current best practices should be followed where possible.

# 2. Executive summary

This report presents the findings of an audit performed by iosiro on the VERA ERC-777 token.

## Audit findings

iosiro noted one low risk and three informational issues. The low risk issue related to minter control. The informational issues related to Solidity version pinning, test coverage, code clarity and gas usage.

## User concerns

Users of the VERA token should note that this is an ERC-777 token. ERC-777 tokens are compatible with ERC-20 tokens. The VERA token has an initial supply of 10,356,466,694. More tokens can be minted by the owner and owner-appointed minters up until the point where a minter calls `finishMinting()` on the token, which will prevent any further minting. This function emits the `MintFinished` event, and the status of minting can be viewed through the view function `isFinishedMinting()`.

The owner and owner-appointed blockers are able to block addresses from sending, receiving and acting as operators for the token. These addresses can be unblocked by the owner and blockers. The view function `isBlockListed(address account)` can be called to check whether an account is blocked.

The token can also be paused and unpaused by the owner or owner-appointed pausers. This will prevent token transfers from taking place.

## Recommendations

At a high level, the security posture of the VERA token could be further strengthened by:

- Remediating the issues identified in this report and performing a review to ensure that the issues were correctly addressed.

- Performing additional audits at regular intervals, as security best practices, tools, and knowledge change over time. Additional audits over the course of the project's lifespan ensure the longevity of the codebase.

- Creating a bug bounty program to encourage the responsible disclosure of security vulnerabilities in the system.

# 3. Audit details

## 3.1 Scope

The source code considered in-scope for the assessment is described below. Code from all other files is considered to be out-of-scope. Out-of-scope code that interacts with in-scope code is assumed to function as intended and introduce no functional or security vulnerabilities for the purposes of this audit.

### 3.1.1 Smart contracts

- **Project name:** VraToken
- **Address:** 0xf411903cbc70a74d22900a5de66a2dda66507255

## 3.2 Methodology

A variety of techniques, described below, were used to conduct the audit.

### 3.2.1 Code review

The source code was manually inspected to identify potential security flaws. Code review is a useful approach for detecting security flaws,

discrepancies between the specification and implementation, design improvements, and high risk areas of the system.

### 3.2.2 Dynamic analysis

The contracts were compiled, deployed, and tested in a Ganache test environment, both manually and through the Truffle test suite provided. Manual analysis was used to confirm that the code operated at a functional level and to verify the exploitability of any potential security issues identified. The coverage report of the provided Truffle tests as on the final day of the audit is given below.

| File | % Sts | % Brnch | % Funcs | % Lines | Missed Lns |
|---|---|---|---|---|---|
| contracts/ | 20 | 13.33 | 19.05 | 20 | |
| ERC777MintablePausableBlocklistable.sol | 35.71 | 18.18 | 30 | 35.71 | 59--101, 118--146 |
| ERC777SenderRecipientMock.sol | 0 | 0 | 0 | 0 | 45--147 |
| VraToken.sol | 100 | 100 | 100 | 100 | |
| All files | 20 | 13.33 | 19.05 | 20 | |

Apart from mocks, for which test coverage is unimportant, test coverage was lacking for the majority of the public functions in

`ERC777MintablePausableBlocklistable`. More detail on the uncovered functionality is provided in 5.4.2 Insufficient test coverage.

### 3.2.3 Automated analysis

Tools were used to automatically detect the presence of several types of security vulnerabilities, including reentrancy, timestamp dependency bugs, and transaction-ordering dependency bugs. The static analysis results were manually analyzed to remove false-positive results. True positive results would be indicated in this report.

Static analysis tools commonly used include Slither, Securify, and MythX. Tools such as the Remix IDE, compilation output, and linters could also be used to identify potential areas of concern.

## 3.3 Risk ratings

Each issue identified during the audit has been assigned a risk rating. The rating is determined based on the criteria outlined below.

- **High risk** - The issue could result in a loss of funds for the contract owner or system users.
- **Medium risk** - The issue resulted in the code specification being implemented incorrectly.
- **Low risk** - A best practice or design issue that could affect the security of the contract.

- **Informational** - A lapse in best practice or a suboptimal design pattern that has a minimal risk of affecting the security of the contract.

- **Closed** - The issue was identified during the audit and has since been addressed to a satisfactory level to remove the risk that it posed.

# 4. Design specification

The following section outlines the intended functionality of the system at a high level. The specification is based on the implementation in the codebase and any perceived points of conflict should be highlighted with the auditing team to determine the source of the discrepancy.

## 4.1 Overview

The VERA token is an ERC-777 token with the following parameters:

| Field | Value |
|---|---|
| Symbol | VRA |
| Name | VERA |
| Decimals | 18 |
| Initial supply | 10,356,466,694.6670751530579940000 |

The Vera token system has the following roles with the following abilities, designated by the contract owner:

- **Pausers**: these users can pause and unpause all token transfers.

- **Minters**: these users can mint tokens. Minters can mint tokens to an account of their choosing until a minter calls the `finishMinting()` function, after which time it will be impossible to mint further tokens. This cannot be undone.

- **Blockers**: these users can add and remove addresses to and from the token's blocklist. Addresses on the blocklist will not be able to send or receive tokens, or to act as operators.

Immediately after the contract is deployed, the owner will be the only Pauser, Minter and Blocker.

# 5. Detailed findings

The following section details the findings of the audit.

## 5.1 High risk

No high risk issues were present at the conclusion of the review.

## 5.2 Medium risk

No medium risk issues were present at the conclusion of the review.

## 5.3 Low risk

### 5.3.1 Minter horizontal privilege escalation

*ERC777MintablePausableBlocklistable*

#### Description

As users with the minter role are permitted to call the `finishMinting()` function, a minter will thus be able to call this function to deny minting abilities to all other minters, including the administrator. This functionality was removed from `ERC20Mintable.sol` in OpenZeppelin for this reason.

### Recommendation

It is recommended that the default administrator role be made the only role that is permitted to run `finishMinting()`.

## 5.4 Informational

### 5.4.1 Inexact Solidity compiler version

*General*

#### Description

The pragma statement included in the smart contracts allowed the codebase to be compiled with any version of Solidity between 0.6.7 and the final release in the 0.6.x series. This could lead to a situation where the contracts are compiled using a compiler version they have not been extensively tested on, or which contains undiscovered bugs.

#### Recommendation

To increase certainty and reliability, it was recommended that the contracts be locked to a specific compiler version. iosiro recommends version 0.6.12.

### 5.4.2 Insufficient test coverage

*ERC777MintablePausableBlocklistable*

#### Description

Certain functionality was found to not be covered by tests. This could impact the maintainability of the code, and increase the likelihood of

introducing functional or security issues into the codebase. The following public functions in `ERC777MintablePausableBlacklistable.sol` were not covered by tests:

- `pause()`
- `unpause()`
- `blockAccount()`
- `unblockAccount()`
- `mint(...)`
- `isFinishedMinting()`
- `finishMinting()`

Additionally, tests did not cover require statement failures in `_beforeTokenTransfer(...)`.

### Recommendation

It is recommended that the test suite be extended to cover the functionality specified above.

## 5.4.3 Design comments

The following actions can be taken to improve the codebase clarity:

- `ERC777MintablePausableBlocklistable`: Comments above the main contract body and contract constructor do not

mention the blocklist functionality. Added this information is recommended to enhance code clarity.

- `ERC777MintablePausableBlocklistable`: The visibility of the `_blocklist` mapping should be defined explicitly; currently it will default to internal.

- `ERC777MintablePausableBlocklistable`: The `pause()`, `unpause()`, `blockAccount(...)`, `unblockAccount(...)`, `mint(...)`, `isFinishedMinting()` and `finishMinting()` functions are not used internally and can be declared `external` to save gas.

## 5.5 Closed

No issues were closed during the audit.