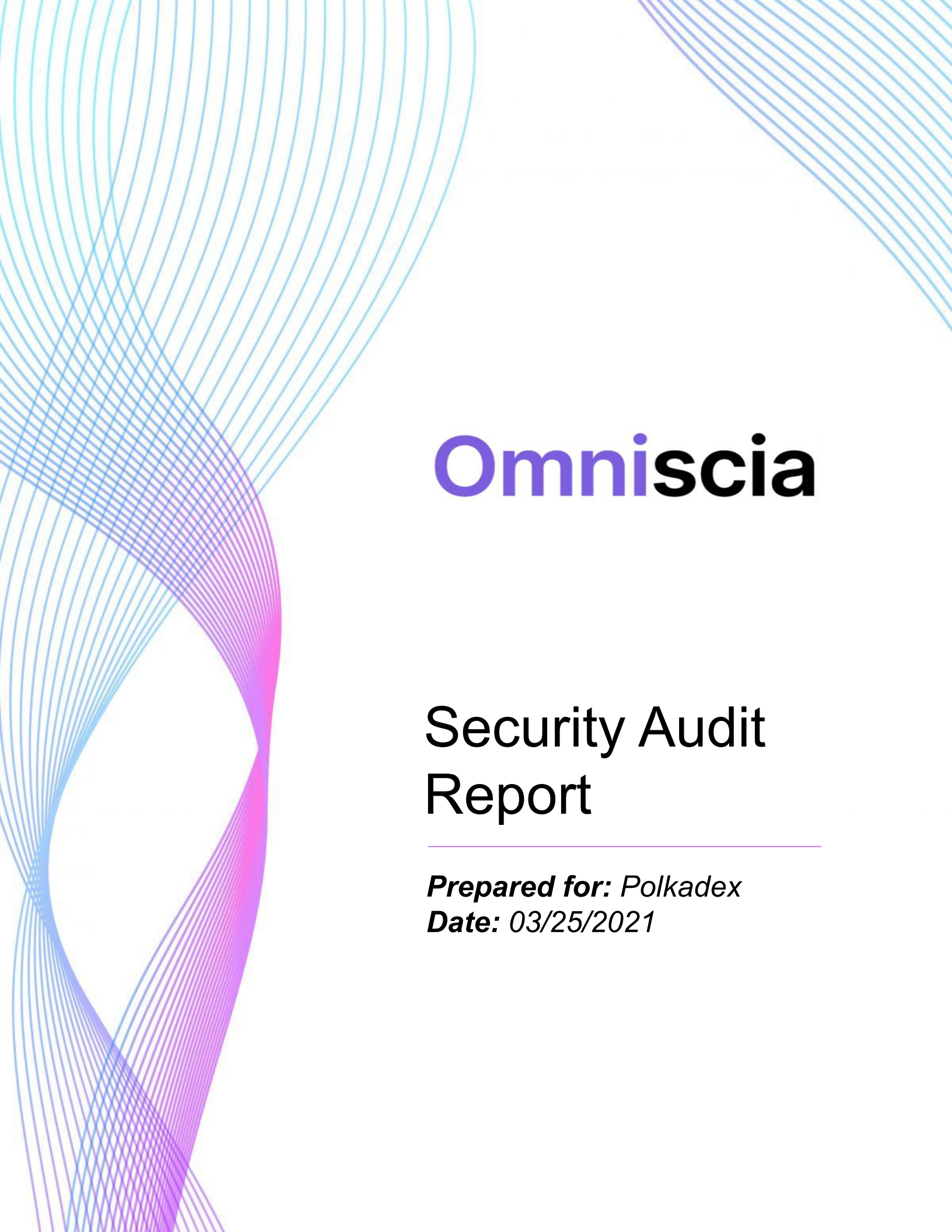

Omniscia

Security Audit Report

Prepared for: *Polkadex*
Date: *03/25/2021*

Polkadex ERC20 Security Review

This document acts as a security review of the Polkadex ERC20 token located at polkadex-ERC20 of the Polkadex-Substrate GitHub account using commit hash 3f38ff05.

Over the course of the audit we ensured that the token implementation is fully compliant with the ERC / EIP standard no. 20 and we validated that all state transitions with regards to the vested balances occur as expected.

While we were able to find minimal issues, in the sake of transparency we would like to point out that the project at hand is fully centralized and contains functions that permit the owner to completely freeze balances at will as well as destroy the token altogether rendering it unusable.

The audit findings are split into two sections; a section dedicated to security findings and a section dedicated to style findings that we believe to be due for the contracts to achieve a production-ready state.

We conducted an alleviation review on commit hash d6273263 and reflected the Polkadex's team actions in each respective alleviation chapter.

Security Findings

This sector contains any finding that would alter the behaviour of the system and thus does not constitute to be `informational` in severity.

PKD-01M: Inconsistent Vesting Establishment

Type	Severity	Location
Indeterminate Code	Minor	dex.sol:L10,L15-L27

Description:

The token is meant to be vested prior to being released to its holders, however, the `MainHolder` function mints the equivalent amount that is being vested directly resulting in a 50%/50% split.

Recommendation:

We advise that if this functionality is desired it is strictly defined so as it is relatively ambiguous in the codebase.

Alleviation:

The Polkadex team evaluated the code and decided to retain its functionality as is.

PKD-02M: Inexistent Zero-Address Check

Type	Severity	Location
Input Sanitization	Minor	dex.sol:L38-L40

Description:

The `TransferOwnerShip` function does not validate that the `newAddress` is non-zero potentially leading to an owner-less contract.

Recommendation:

We advise that a zero-check is introduced in the function that evaluates the `newAddress` variable to not be equal to the zero-address via a `require` check.

Alleviation:

A zero-address `require` check was introduced to the `TransferOwnership` function properly.

PKD-03M: Redundant Centralization

Type	Severity	Location
Logical Issue	Medium	dex.sol

Description:

The contract permits its `Owner` to freeze it at will as well as completely remove it from existence, rendering it incompatible with centralized exchanges as well as potentially ill-received by the community.

Recommendation:

We advise that the centralization level desired is greatly considered as at its current state the token should not exist as a smart contract and completely defeats the purposes of being so.

Alleviation:

The functionality was removed from the contract properly decentralizing the transact-ability aspect of the token.

Optimization & Style Findings

This section includes all style guidance findings as well as any gas optimizations that can be applied to the codebase using aggressive techniques such as tight-packing and more.

ERC-01S: Inconsistent Variable Naming Conventions

Type	Severity	Location
Code Style	Informational	ERC20.sol:L255,L261,L271-L274,L275,L276,L277,L279,L633

Description:

The official Solidity documentation contains a section dedicated to a uniform Style Guide that aims to make code as legible as possible.

The linked segments do not conform to this convention as their capitalization and overall naming paradigm is not compliant with the standard.

Recommendation:

We advise that the naming variables are adjusted to properly reflect the Solidity naming convention guidelines and render the code more legible.

Alleviation:

The Polkadex team considered our recommendations but decided to retain the current convention as is citing time constraints and personal preference.

ERC-02S: Variable Mutability Optimization

Type	Severity	Location
Gas Optimization	Informational	ERC20.sol:L269,L329

Description:

The `_decimals` variable is solely assigned to a value literal during the contract's `constructor`.

Recommendation:

We advise that the assignment is instead relocated to the `_decimals` declaration allowing it to be declared `constant` and thus greatly optimize the gas cost involved in utilizing it.

Alleviation:

The Polkadex team considered our recommendations but decided to not apply them citing time constraints.

PKD-01S: Inconsistent Variable Naming Conventions

Type	Severity	Location
Code Style	Informational	dex.sol:L6,L7,L15,L29,L34,L38,L42

Description:

The official Solidity documentation contains a section dedicated to a uniform Style Guide that aims to make code as legible as possible.

The linked segments do not conform to this convention as their capitalization and overall naming paradigm is not compliant with the standard.

Recommendation:

We advise that the naming variables are adjusted to properly reflect the Solidity naming convention guidelines and render the code more legible.

Alleviation:

The Polkadex team considered our recommendations but decided to retain the current convention as is citing time constraints and personal preference.

PKD-02S: Variable Mutability Optimization

Type	Severity	Location
Gas Optimization	Informational	dex.sol:L7,L12

Description:

The `InitialBlockNumber` variable is solely assigned to during the contract's `constructor`.

Recommendation:

We advise that the `immutable` variable mutability specifier is introduced to the variable to significantly reduce the gas cost involved in interacting with it. Additionally, we advise visibility specifiers to be set across both contracts wherever unset to ensure no ambiguity can arise from compiler differences.

Alleviation:

The variable was properly set as `immutable` greatly optimizing the gas cost of wherever it is utilized in.