MixBytes()

PIEDAO

VESTED-TOKEN-MIGRATION AND CRUST SMART CONTRACTS AUDIT REPORT

SEPTEMBER 15
2020

FOREWORD
TO REPORT

A small bug can cost you millions. **MixBytes** is a team of experienced blockchain engineers that reviews your codebase and helps you avoid potential heavy losses. More than 10 years of expertise in information security and high-load services and 18 000+ lines of audited code speak for themselves.

This document outlines our methodology, scope of work, and results.

We would like to thank **PieDAO** for their trust and opportunity to audit their smart contracts.

CONTENT
DISCLAIMER

This report was made public upon consent of **PieDAO**. **MixBytes** is not to be held responsible for any damage arising from or connected with the report.

Smart contract security audit does not guarantee a comprehensive inclusive analysis disclosing all possible errors and vulnerabilities but covers the majority of issues that represent threat to smart contract operation, have been overlooked or should be fixed.

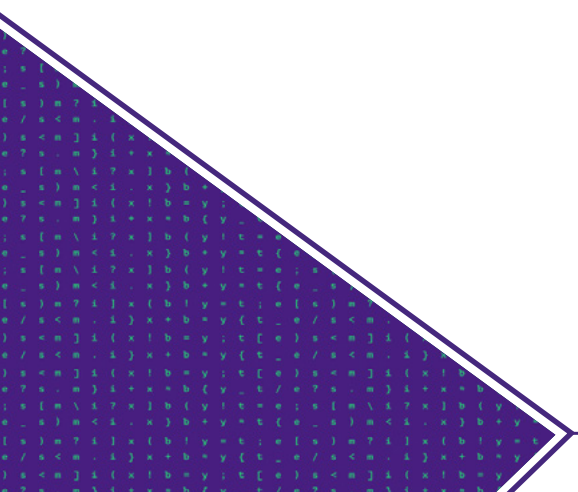

TABLE OF CONTENTS

01 INTRODUCTION TO THE AUDIT

GENERAL PROVISIONS

PieDAO is an asset allocation DAO for decentralized market-weighted portfolio allocations.

MixBytes was approached by **PieDAO** to provide a security assessment of a part of vested token migration and "Crust" token contracts.

SCOPE OF THE AUDIT

AUDITED OBJECT	LOCATION
Smart contracts	**vested-token-migration-app**
	pie-crust

02 SECURITY ASSESSMENT PRINCIPLES

CLASSIFICATION OF ISSUES

CRITICAL

Bugs leading to Ether or token theft, fund access locking or any other loss of Ether/tokens to be transferred to any party (for example, dividends).

MAJOR

Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.

WARNINGS

Bugs that can break the intended contract logic or expose it to DoS attacks.

COMMENTS

Other issues and recommendations reported to/acknowledged by the team.

SECURITY ASSESMENT METHODOLOGY

Two auditors independently verified the code.

Stages of the audit were as follows:

1. «Blind» manual check of the code and its model
2. «Guided» manual code review
3. Checking the code compliance to customer requirements
4. Discussion of independent audit results
5. Report preparation

03 DETECTED ISSUES

CRITICAL

Not found.

MAJOR

1.VestedTokenMigration.sol#L91

The return value can incorrectly excess `_amount` that will result in redundant token migration (more than `_windowAmount`). The statement **VestedTokenMigration.sol#L72** won't help because it was applied too late - after subtracting `amountMigratedFromWindow[leaf]`. It means that several transactions of `_amount` less or equal to `_windowAmount` will succeed.

Proof of concept:
https://gist.github.com/Eenae/dc83467d4adb6c8667c768af1bd0b0b4

The code simulates a moment way ahead of `windowVestingEnd`. After deployment of the Test contract in Remix we'll be able to make several `migrateVested` calls with the `_amount` equal to 100 from the same account. Each call will emit a `Migrated` event with the `_migratedAmount` equal to 100, meaning that the migration was successful and the tokens were minted. Also note that `amountMigratedFromWindow` will yield a value greater than 100 after the second migration.

The issue is not marked as critical since actual over-migration of tokens is unlikely thanks to burning of existing tokens here: **VestedTokenMigration.sol#L76**.

We suggest adding an `if (_time >= _vestingEnd) return _amount;` statement to the `calcVestedAmount` function. Also, the line **VestedTokenMigration.sol#L72** will become obsolete.

Status:

FIXED - at **2ebb401**

WARNINGS

1. Crust.sol#L67
Crust.sol#L87

Differences in the `crumbs` decimals are not taken into account during summation. Moreover, `decimals` of the `Crust` can be arbitrary, that, in turn, can lead to the `crumbs` token domination over the entire `Crust`.

Take two tokens T1 and T2 as an example. Let `T1.decimals = 10`, `T2.decimals = 4` and `Crust.decimals = 10`. Addinging 1 full token of T1 to 1 full token of T2 will give `uint256: 10000010000` that is roughly equal to 1 full Crust token, i.e. T2 contribution was almost ignored. Please note that the current solution doesn't implement the stated objective "each token has the same weight", quite the opposite. Decimal field values are implementation details and should not influence the outcome.

To achieve the stated objectives and get the `crumbs` to the same scale, we suggest using linear normalization technique.

`Crust.decimals` should be computed as `max(Ti.decimals())` for each token `i` in the `crumbs`.

After normalization, our example will result in `1e10 * 10**(10-10) + 1e4 * 10**(10-4) = 2e10`, i.e. 2 full Crust tokens.

Status:

FIXED - at **f2400b5**

COMMENTS

1. VestedTokenMigration.sol#L69

We suggest returning zero from the function as soon as `migrateAmount` equals zero.

Status:

FIXED - at **2ebb401**

2. VestedTokenMigration.sol#L74

An assertion `assert(amountMigratedFromWindow[leaf] <= _windowAmount);` could be added.

Status:

FIXED - at 2ebb401

3. VestedTokenMigration.sol#L43

It's allowed to change the already set merkle root. Make sure that this is desired behaviour.

Status:

ACKNOWLEDGED

4. VestedTokenMigration.sol#L88

We recommend adding a strict check `_vestingStart < _vestingEnd` here to ensure that the function always works with the correct input that in turn will reduce the number of input data invariants.

Status:

FIXED - at 2ebb401

5. VestedTokenMigration.sol#L30
VestedTokenMigration.sol#L42
Crust.sol#L18

We recommend introducing a check to ensure the parameters are not equal to zero.

Status:

FIXED - at cce403c

6. Crust.sol#L25-L47

For each public state variable an automatic getter function is generated. This means getters for name, symbol and decimals may be removed. See **this** for details.

Status:

FIXED - at **c2fbe8f**

04 CONCLUSION AND RESULTS

Provided smart contracts were audited and several troublesome issues were identified:

* no critical issues
* 1 major issue
* and also several recomendational comments

All issues were fixed and following commits don't have any vulnerabilities according to our analysis:

* **https://github.com/pie-dao/vested-token-migration-app/commit/779a9e1f7636df675323034a8196f430c5a91102**
* **https://github.com/pie-dao/pie-crust/commit/f2400b5422e1ad4fb45253a6f0ff4ea9102cf0af**

ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build open-source solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, consult universities and enterprises, do research, publish articles and documentation.

Stack

Blockchains

JOIN US