TOSDIS FINANCE STAKEMASTER SMART CONTRACT AUDIT

March 01, 2021

MixBytes()

CONTENTS

1. INTRODUCTION

1.1 DISCLAIMER

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, 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 information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of TosDis Finance. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 PROJECT OVERVIEW

TosDis is a p2p lending platform that runs on Ethereum-based Smart Contracts to create a safe and efficient environment where borrowers worldwide have fast and convenient access to loans, and lenders can find high-yield investment opportunities.

1.3 SECURITY ASSESSMENT METHODOLOGY

At least 2 auditors are involved in the work on the audit who check the provided source code independently of each other in accordance with the methodology described below:

01 "Blind" audit includes:
> Manual code study
> "Reverse" research and study of the architecture of the code based on the source code only

Stage goal:
Building an independent view of the project's architecture
Finding logical flaws

02 Checking the code against the checklist of known vulnerabilities includes:
> Manual code check for vulnerabilities from the company's internal checklist
> The company's checklist is constantly updated based on the analysis of hacks, research and audit of the clients' code

Stage goal:
Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flashloan attacks, etc.)

03 Checking the logic, architecture of the security model for compliance with the desired model, which includes:
> Detailed study of the project documentation
> Examining contracts tests
> Examining comments in code
> Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of the reports from all auditors into one common interim report document
> Cross check: each auditor reviews the reports of the others
> Discussion of the found issues by the auditors
> Formation of a general (merged) report

Stage goal:
Re-check all the problems for relevance and correctness of the threat level
Provide the client with an interim report

05 Bug fixing & re-check.
> Client fixes or comments on every issue
> Upon completion of the bug fixing, the auditors double-check each fix and set the statuses with a link to the fix

Stage goal:
Preparation of the final code version with all the fixes

06 Preparation of the final audit report and delivery to the customer.

Findings discovered during the audit are classified as follows:

FINDINGS SEVERITY BREAKDOWN

Level	Description	Required action
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party	Immediate action to fix issue
Major	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.	Implement fix as soon as possible
Warning	Bugs that can break the intended contract logic or expose it to DoS attacks	Take into consideration and implement fix in certain period
Comment	Other issues and recommendations reported to/acknowledged by the team	Take into consideration

Based on the feedback received from the Customer's team regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The project team is aware of this finding. Recommendations for this finding are planned to be resolved in the future. This finding does not affect the overall safety of the project.
No issue	Finding does not affect the overall safety of the project and does not violate the logic of its work.

1.4 EXECUTIVE SUMMARY

The audited scope implements custom-token staking pools creation and management. Such staking pools are fixed-supply pools and distribute the reward according to the amount of time the particular participant had it's funds staked. Fees for pool management are being payed in deflationary token.

1.5 PROJECT DASHBOARD

Client	TosDis Finance
Audit name	StakeMaster
Initial version	be50dbf8a52a8f919694498bf30394d328d88fbb
Final version	2e4aad91a77318a1134fa1562031f9634b46e705
SLOC	285
Date	2021-02-15 - 2021-03-01
Auditors engaged	2 auditors

FILES LISTING

FeeToken.sol	FeeToken.sol
StakeMaster.sol	StakeMaster.sol
StakingPool.sol	StakingPool.sol
ERC20Basic.sol	ERC20Basic.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	2
Warning	3
Comment	2

CONCLUSION

Smart contracts have been audited and several suspicious places have been spotted. During the audit two issues were marked as major because they could lead to some undesired behavior, also several warnings and comments were found and discussed with the client. After working on the reported findings all of them were resolved or acknowledged (if the problem was not critical). So, the contracts are assumed as secure to use according to our security criteria. Final commit identifier with all fixes: `2e4aad91a77318a1134fa1562031f9634b46e705`.

2. FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

MJR-1	Wrongly increased `participants` amount
File	StakingPool.sol
Severity	Major
Status	Fixed at 2e4aad91

DESCRIPTION

At line StakingPool.sol#L120 contract increases participants amount, that happens if `user.amount` is zero, however if user will provide zero `_amountToStake` `participants` also will increased. So somebody can increase `participants` infinitely by calling `stakeTokens` with zero stake.

RECOMMENDATION

We recommend increase `participants` only if `_amountToStake` not zero

MJR-2	Potentially differentiating message senders
File	FeeToken.sol
Severity	Major
Status	Fixed at 2e4aad91

DESCRIPTION

This warning is about inconsistent message sender address source (`msg.sender` and `_msgSender()`) usage in here: FeeToken.sol#L37, FeeToken.sol#L39, FeeToken.sol#L40.

This can lead to the incorrect message sender address choice in case, for example, particular function call was a relayed call or meta-transaction-related call.

RECOMMENDATION

It is recommended to switch to some particular message sender address retrieval method to avoid double-ownership (or misownership). In particular, using `AccessContolList` is better to choose `_mseSender()`.

2.3 WARNING

WRN-1	Non-explicit multiplier computation checks
File	StakingPool.sol
Severity	Warning
Status	Fixed at 2e4aad91

DESCRIPTION

This warning is about potentially faulty implicit checks in `getMultiplier` function in here: StakingPool.sol#L66.

This function seems to suppose to always return some valid value (even if particular input arguments are not valid), and business logic implicitly supposes that as well. Unfortunately, `getMultiplier` function returns valid value `0` in case `_from` and `_to` arguments are incorrect in relation to `finishBlock` variable, but does not return any valid value in case `_from` and `_to` arguments are incorrect among themselves. This can lead to the business logic implicitly assuming `getMultiplier` functions always returns something valid failure, for example, in here: StakingPool.sol#L104.

RECOMMENDATION

It is recommended to either consider `getMultiplier` function returning valid values for every case (`0` for all the incorrect arguments cases for example), or refactor the application logic to support this function unwinding.

WRN-2	Keep invariant `burnPercent <= divider`
File	StakeMaster.sol
Severity	Warning
Status	Fixed at 2e4aad91

DESCRIPTION

According logic at line StakeMaster.sol#L75 `burnPercent` should be less or equal than `divider`, but in contract that invariant never checked

RECOMMENDATION

We recommend add particular check

WRN-3	Restrict deflationary tokens
File	StakingPool.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At line StakingPool.sol#L124 contract receives token by `safeTransferFrom`, and increases `user.amount` by `_amountToStake`, however that approach doesn't work with deflationary tokens because in that case real received amount will be less than requested. That will break the whole logic of contract if somebody creates pool with deflationary token.

RECOMMENDATION

We recommend to support such tokens or restrict contract usage with them

CLIENT'S COMMENTARY

We'll add notice to UI that deflationary and rebase tokens are not supported.

2.4 COMMENTS

CMT-1	Misleading docs
File	FeeToken.sol
Severity	Comment
Status	Fixed at 2e4aad91

DESCRIPTION

This comment is about misleading documentation mentioning there is a `PAUSER_ROLE` (and there is no such a role in this particular case) in here: FeeToken.sol#L27.

RECOMMENDATION

It is recommended to remove misleading description.

CMT-2	Unrestricted emergency withdrawal
File	StakingPool.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

This comment is about unrestricted emergency withdrawal function in here: StakingPool.sol#L170 being possible to be called anytime with burning all the rewards being earned.

RECOMMENDATION

Since it seems to be fine to make user reponsible to choose, to call the function retrieving staked funds with or without rewards being taken care of, more appropriate solution seems to be to introduce a so-called "Emergency mode" (with introducing a function modifier to check if that emergency has happened), initiated by some governance members. Probably only in case of emergency such a reward-burning withdrawal should be available.

3. 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, do research and tech consultancy.

BLOCKCHAINS

 Ethereum

 Cosmos

 EOS

 Substrate

TECH STACK

 Python

 Solidity

 Rust

 C++

CONTACTS

 https://github.com/mixbytes/audits_public

 https://mixbytes.io/

 hello@mixbytes.io

 https://t.me/MixBytes

 https://twitter.com/mixbytes