MixBytes()

DEFISTARTER

SMART CONTRACT

AUDIT REPORT

NOVEMBER 06

2020

TABLE OF CONTENTS

01 INTRODUCTION TO THE AUDIT

GENERAL PROVISIONS

DeFiStarter is a decentralized crowdfunding platform that provides a unique opportunity to reduce investors' risks to zero, and to allow startups to raise financing without directly selling their tokens.

SCOPE OF AUDIT

https://github.com/defistarter/contracts/tree/9ffe009ee6047ced669711dde14fe38483abdf7e excluding `token-helpers` directory.

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 ASSESSMENT 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. Potential claiming stuck for several periods after withdraw

StakingPool.sol#L341

Here we have division by `periodTotalSupply`, in case if `periodTotalSupply` becomes zero that it will be impossible to claim any reward.

Attack flow:

Assuming that we have 1 user and 5 periods:

* at period 0: user stake 10 tokens (`_historyTotalSupply[0] = 10`, `user.period = 0`)
* at period 1: user claimReward and withdraw all 10 tokens (`_historyTotalSupply[1] = 0`, `user.period = 1`)
* at period 2: do nothing (`_historyTotalSupply[2] = 0`, `user.period = 1`)
* at period 3: user stake 20 tokens (`_historyTotalSupply[3] = 20`, `user.period = 1`)
* at period 4: user try to `claimReward` and get error “SafeMath: division by zero”
* contract state at period 4 is:
  * `_historyTotalSupply = [10, 0, 0, 20, 0]`
  * `user.period = 1`

So if user try to claimReward we will get:

* `savedTotalSupply` = `0` (**StakingPool.sol#L326**)
* `periodTotalSupply` = `0` (**StakingPool.sol#L333**)

and finally here we have division by zero here: **StakingPool.sol#L341**

We recommend to check the `periodTotalSupply` value before division and omit that in case of zero.

This particular issue is not marked as critical because that cannot be exploited to steal funds, but issues should be fixed ASAP due to that can break desired contract logic.

Status:

FIXED at **56377b4** and **6fc21c4**

WARNINGS

1. Potential inability to claim reward

StakingPool.sol#L328

There is a `for` loop with potentially huge amounts of iterations, so that can take an uncontrolled amount of gas and that can cause inability to calculate reward amount for particular users if gas consumption reaches the maximum limit.

We recommend strictly limiting the amount of loop iterations and allow users to claim their rewards partially in several batches.

Status:

ACKNOWLEDGED

2. `claimReward`/`withdraw` allowed to execute before staking starts

StakingPool.sol#L259

StakingPool.sol#L245

These methods formally can be executed before staking starts but after setup.

It's not critical in that case, but anyway we recommend explicitly checking current contract status to minimize the number of possible invariants.

Status:

FIXED at 51d1ded

COMMENTS

1. Unoptimized check in `calculateReward` loop

StakingPool.sol#L331

```
if(i > user.period){
 …
}
```

That check always returns true after the first iteration, so it spent excess gas each iteration. We recommend moving the first iteration calculations out of loop and start loop from `user.period + 1`.

Status:

FIXED at 56377b4

2. Different code style

StakingPool.sol#L105

Here used snake case for `_fee_beneficiary` naming, but whole other code uses camel case. We recommend using the same naming condition.

Status:

FIXED at 56377b4

3. Typo in modifier name

StakingPool.sol#L68

Missed character in `onlyAferSetup`.

Status:

FIXED at 56377b4

4. Too complicated status checks

StakingPool.sol#L135

StakingPool.sol#L148

StakingPool.sol#L161

StakingPool.sol#L235

StakingPool.sol#L265

There are a lot of different status checks, some of them directly using `endTime`, `closeTime`, `startTime` mixed with `status` field.

So it can cause potential issues in the future related to incorrect status interpretation.

We recommend defining all possible statutes to special `Status` enum and move checks to particular modifiers as `onlyAfterSetup`.

Status:

ACKNOWLEDGED

04 | CONCLUSION AND RESULTS

Finding list

Level	Amount
CRITICAL	-
MAJOR	1
WARNING	2
COMMENT	4

All critical and major issues have been fixed.

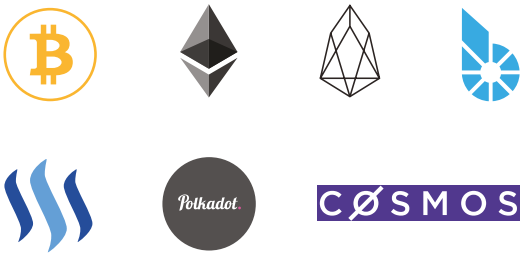
Final commit identifier with fixes: `07edefa4c5931e5fbef97c61d48062c606fc21c4`

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