KEEP3R STAKING REWARDS SMART CONTRACT AUDIT

November 11, 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 KEEP3R NETWORK. 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 SECURITY ASSESSMENT METHODOLOGY

A group of 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 Project architecture review:
> Reviewing project documentation
> General code review
> Reverse research and study of the architecture of the code based on the source code only
> Mockup prototyping

Stage goal:
Building an independent view of the project's architecture and identifying logical flaws in the code.

02 Checking the code against the checklist of known vulnerabilities:
> 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
> Checking with static analyzers (i.e Slither, Mythril, etc.)

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

03 Checking the code for compliance with the desired security model:
> 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
> Exploits PoC development using Brownie

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of interim auditor reports into a general one:
> 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 and 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.

1.3 PROJECT OVERVIEW

Keep3r Network is a decentralized keeper network for projects that need external devops and for external teams to find keeper jobs.StakingRewardsV3 allows liquidity providers of the Uniswap V3 pools deposit their NFT (which represents active position in pool) via `deposit()` function. After that users can wait some time to accumulate rewards on their NFT and return token via `withdraw()` function. Accumulated rewards can be gotten from StakingRewardsV3 smart contract via `getRewards()` function. It is necessary to mention that when users deposit their NFT to contract, fees, accumulated on their NFT, go to contract owner. In exchange users can get special reward token from contract.

1.4 PROJECT DASHBOARD

Client	KEEP3R NETWORK
Audit name	Staking Rewards
Initial version	13ecc6966ae1a413f62224382bfd4d64b1a22351
Final version	7ba64a6c537b83690785ee740ebc0beb4f154811
Date	October 06, 2021 - November 11, 2021
Auditors engaged	5 auditors

FILES LISTING

StakingRewardsV3-1.sol	https://github.com/keep3r-network/StakingRewardsV3/tree/13ecc6966ae1a413f62224382bfd4d64b1a22351/contracts/StakingRewardsV3-1.sol

FINDINGS SUMMARY

Level	Amount
Critical	2
Major	2
Warning	4
Comment	11

CONCLUSION

Smart contract has been audited and several suspicious places have been spotted. During the audit 2 critical issues were found and reported to the client. 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) by the client.Final commit identifier with all fixes: `7ba64a6c537b83690785ee740ebc0beb4f154811`

2.FINDINGS REPORT

2.1 CRITICAL

CRT-1	Impossible withdraw for smart contract
File	StakingRewardsV3-1.sol
Severity	Critical
Status	Fixed at 7ba64a6c

DESCRIPTION

If any smart contract deposits NFT to StakingRewardsV3 it must have `onERC721Received()` function or `withdraw()` will always revert:
StakingRewardsV3-1.sol#L256

RECOMMENDATION

We recommend to use `transferFrom()` instead of `safeTransferFrom()`.

CRT-2	Incorrect update of `totalLiquidity`
File	StakingRewardsV3-1.sol
Severity	Critical
Status	**Fixed** at **7ba64a6c**

DESCRIPTION

If user calls `deposit()` -> `withdraw()` -> `getReward()` then contract will incorrectly calculate `totalLiquidity` which will lead to incorrect calculations of rewards for users: StakingRewardsV3-1.sol#L342

RECOMMENDATION

We recommend to change the logic of `update` modificator, so that`totalLiquidity` would update only if NFT is possessed to this contract.

2.2 MAJOR

MJR-1	Incorrect calculation of `rewardPerLiquidity`
File	StakingRewardsV3-1.sol
Severity	Major
Status	**Acknowledged**

DESCRIPTION

If the first user deposits NFT after some time from `notify()` call, then `(lastTimeRewardApplicable() - lastUpdateTime)` always will be less than `DURATION` which leads to freezing some funds on the contract:
StakingRewardsV3-1.sol#L156

RECOMMENDATION

We recommend to change the calculation of `rewardPerLiquidity`.

CLIENT'S COMMENTARY

Acceptable as it only locks rewards, not user funds

MJR-2	Possible ddos attack
File	StakingRewardsV3-1.sol
Severity	Major
Status	**Fixed** at **7ba64a6c**

DESCRIPTION

Malicious user can front run `withdraw()` function to change the current price in pool, so user can lost all his rewards:
StakingRewardsV3-1.sol#L195

RECOMMENDATION

We recommend to get an average price for this check.

2.3 WARNING

WRN-1	Addresses not checked
File	StakingRewardsV3-1.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

Input addresses are not checked:
StakingRewardsV3-1.sol#L139

RECOMMENDATION

We recommend to add a check that input addresses are not equal to zero address.

WRN-2	Impossible situation
File	StakingRewardsV3-1.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

`_index >= _length` can't be `true`:
StakingRewardsV3-1.sol#L236

RECOMMENDATION

We recommend to call `revert` if `_index >= _length` is equal to `true`.

WRN-3	`_lastUpdateTime` can be equal to zero
File	StakingRewardsV3-1.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

`_lastUpdateTime` can be equal to zero if user deposits NFT before the first call of `notify`:
StakingRewardsV3-1.sol#L337

RECOMMENDATION

We recommend to add a check that user can't deposit before the first call of `notify`.

WRN-4	Input parameters in `notify()` not checked
File	StakingRewardsV3-1.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

`notify()` can be called with `amount` equal to zero:
StakingRewardsV3-1.sol#L299

RECOMMENDATION

It is recommended to add `require(amount > 0, "Incorrect input data")` in function `notify()`.

2.4 COMMENT

CMT-1	Function not used
File	StakingRewardsV3-1.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

Function `max()` is not used in the contract:
StakingRewardsV3-1.sol#L9

RECOMMENDATION

We recommend to remove this function.

CMT-2	Not enough comments
File	StakingRewardsV3-1.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

All storage variables don't have comments, so it is harder to understand the code: StakingRewardsV3-1.sol#L99

RECOMMENDATION

We recommend to add comments for all storage variables.

CMT-3	Visibility not set
File	StakingRewardsV3-1.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

Visibility is not set explicitly for some storage variables:
StakingRewardsV3-1.sol#L102

RECOMMENDATION

We recommend to explicitly set visibility for all storage variables.

CMT-4	`nonReentrant` modificator not used
File	StakingRewardsV3-1.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

All functions which can be called by user don't have `nonReentrant` modificator:
StakingRewardsV3-1.sol#L208

RECOMMENDATION

We recommend to add the `nonReentrant` modificator for each function which can be called by user (`deposit()`, `withdraw()`, `getRewards()`) to increase security of the contract.

CMT-5	`require` without message
File	StakingRewardsV3-1.sol StakingRewardsV3-1.sol
Severity	Comment
Status	**Acknowledged**

DESCRIPTION

Here `require` does not use the message, so it is impossible to distinguish them:
StakingRewardsV3-1.sol#L211
StakingRewardsV3-1.sol#L250
StakingRewardsV3-1.sol#L300

RECOMMENDATION

We recommend to add the message to `require`.

CMT-6	Meaningless function
File	StakingRewardsV3-1.sol
Severity	Comment
Status	**No Issue**

DESCRIPTION

Meaning of this function is unclear, because it only calls `notify()` with weird check of unused parameter:
StakingRewardsV3-1.sol#L294

RECOMMENDATION

We recommend to remove this function.

CLIENT'S COMMENTARY

This function is added for compatibility with another already deployed contract, GaugeProxy

CMT-7	Rewrite `withdraw()` for saving gas
File	StakingRewardsV3-1.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

In `withdraw()` function each call of the `withdraw(_tokens[i])` would trigger call of the `update` modificator:
StakingRewardsV3-1.sol#L287

RECOMMENDATION

It is recommended to rewrite `withdraw()` function for saving some gas.

CMT-8	Changing the contract owner is not possible
File	StakingRewardsV3-1.sol
Severity	Comment
Status	**Fixed** at **7ba64a6c**

DESCRIPTION

The `owner` parameter has the modificator `immutable` that locks any modifications after `constructor()` is called.
StakingRewardsV3-1.sol#L116

RECOMMENDATION

It's not necessary, but we recommend to add the owner changing function.

CLIENT'S COMMENTARY

Switched to use setGov/acceptGov in latest commits

CMT-9	Some gas save in `getRewards()` function
File	StakingRewardsV3-1.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

Inside the loop of the `getRewards` call there is a call of the `getReward` function. This function has `update` modifier that updates state variables every time.
StakingRewardsV3-1.sol#L264
It's ok if we make single call `getReward(tokenId)` in transaction, but few calls in the loop will do the same things many times. Actually we need to run this code only once.

RECOMMENDATION

We recommend to refactor function `getRewards()` for getting away of unnecessary and repeatting state modification in the loop.

CMT-10	Get rewards on withdraw
File	StakingRewardsV3-1.sol
Severity	Comment
Status	**No Issue**

DESCRIPTION

User has to make the second call of `getReward` before or after `withdraw(tokenId)`:
StakingRewardsV3-1.sol#L257

RECOMMENDATION

We recommend to add `getReward(tokenId)` inside `_withdraw(tokenId)`.

CLIENT'S COMMENTARY

Withdraw excludes getReward in case a situation occurs where rewards can't be claimed.

CMT-11	Batch processing in the `collect`
File	StakingRewardsV3-1.sol
Severity	Comment
Status	**Fixed** at **7ba64a6c**

DESCRIPTION

The smart contract likely owns multiple `tokenIds` which should be `collect`-ed. We can batch that job to save transaction number and gas:
StakingRewardsV3-1.sol#L159

RECOMMENDATION

We recommend implementing the `collect` function to take array of `tokenId` as an argument.

CLIENT'S COMMENTARY

Implemented in `7ba64a6c537b83690785ee740ebc0beb4f154811`

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