NUCYPHER POOLING STAKING CONTRACTV2 SMART CONTRACT AUDIT

March 25, 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 NuCypher. 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

The audited scope implements part of a decentralized network for secrets management and dynamic access control.

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 contract implements custom staking pool protocol which manage deposits and rewards.

1.5 PROJECT DASHBOARD

Client	NuCypher
Audit name	PoolingStakingContractV2
Initial version	436ae0f134255fabcd49a1d6b5b1eae4fd8c9d51
Final version	afd803d535bafaea26d2fe67e408a42b0608abef
SLOC	192
Date	2021-03-01 - 2021-03-25
Auditors engaged	2 auditors

FILES LISTING

PoolingStakingContractV2.sol	PoolingStakingContrac...

FINDINGS SUMMARY

Level	Amount
Critical	1
Major	1
Warning	3
Comment	10

CONCLUSION

Smart contracts have been audited and several suspicious places were found. During audit one critical and one major issues were identified as they could lead to some undesired behavior also several issues were marked as warning and comments. After working on audit report all issues were fixed or acknowledged(if issue is not critical or major) by client or concluded as not an issue.Final commit identifier with all fixes: `afd803d535bafaea26d2fe67e408a42b0608abef`

2. FINDINGS REPORT

2.1 CRITICAL

CRT-1	Reward sniffing
File	PoolingStakingContractV2.sol
Severity	Critical
Status	Fixed at afd803d5

DESCRIPTION

User can deposit-withdraw tokens several times (at the same transaction), causing reward sniffing.
The emulation of this behavior is presented below:

```
def main():
    deployer = accounts[0]
    workerOwner = accounts[1]

    escrow = deployer.deploy(StakingEscrowMock)
    escrow.setAllTokens(9000)

    token = deployer.deploy(EasyToken, 1000_000)
    stacking = deployer.deploy(PoolingStakingContractV2)
    workerFraction = 1
    stacking.initialize(workerFraction, token, escrow, workerOwner, {'from': deployer})
    stacking.enableDeposit({'from': deployer})

    user1 = accounts[2]
    user2 = accounts[3]
    deployer.transfer(stacking, 6000)
    token.mint(user1, 1000_000)
    token.mint(user2, 1000_000)

    token.approve(stacking, 100_000, {'from': user1})
    stacking.depositTokens(100_000, {'from': user1})

    for _ in range(100):
        token.approve(stacking, 100, {'from': user2})
        stacking.depositTokens(100, {'from': user2})
        stacking.withdrawAll({'from': user2})

    user1_balance = token.balanceOf(user1)
    user2_balance = token.balanceOf(user2)
    stacking_balance = token.balanceOf(stacking)

    print("user1_balance", user1_balance)
    print("user2_balance", user2_balance)
    print("stacking_balance", stacking_balance)
```

```
    stacking.withdrawAll({'from': user1})  # >>>> ERROR HERE <<<<
```

RECOMMENDATION

May be add a check in withdrawAll function to requre deposit DISABLED?

2.2 MAJOR

MJR-1	Reentry in `withdrawAll`
File	PoolingStakingContractV2.sol
Severity	Major
Status	No issue

DESCRIPTION

Malicious token/workerOwner may have a reentry callback to the contract here

PoolingStakingContractV2.sol#L236

but the state of the contract changes here

PoolingStakingContractV2.sol#L247
it allows to use reentry.

RECOMMENDATION

Put transfers as the last statements of the method.

CLIENT'S COMMENTARY

> Token contract is trustable contract, all calls are safe.

2.3 WARNING

WRN-1	Lack of docs
File	PoolingStakingContractV2.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

The purpose of the contract and the way it will work is not clear from the code. Also the logic with accumulated `getCumulativeReward` and `totalWithdrawnReward` must be described in the code as well as the typical scenario of the contract usage.

RECOMMENDATION

Add comprehensive docstrings.

WRN-2	Reentry causing events misordering
File	PoolingStakingContractV2.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At lines:

- PoolingStakingContractV2.sol#L105
- PoolingStakingContractV2.sol#L192
- PoolingStakingContractV2.sol#L211
- PoolingStakingContractV2.sol#L236
- PoolingStakingContractV2.sol#L249
- PoolingStakingContractV2.sol#L255
- PoolingStakingContractV2.sol#L287

RECOMMENDATION

Place the `transfer` on the last line of the method.

CLIENT'S COMMENTARY

> All contracts that are called from pool are trustable, transfers of ETH are tested for reentrancy.

WRN-3	Deflation tokens support
File	PoolingStakingContractV2.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

It might be never known that after calling `transfer(value)` the token will really increase someone's balance by `value`, some deflation tokens "burn" some value on every transfer call. So the only way to know it is explicitly checking balance of the tokens' owner after transfer.

RECOMMENDATION

Add deflation tokens support or explicitly specify in docstring that deflation tokens are not supported.

2.4 COMMENTS

CMT-1	Use different tokens as deposit and rewards
File	PoolingStakingContractV2.sol
Severity	Comment
Status	No issue

DESCRIPTION

It's a common practice to split deposit tokens and rewards tokens into 2 different types of tokens because it's safer and user can always withdraw his deposit. Also, attacks on rewards would not affect the deposit itself.

RECOMMENDATION

We recommend to split tokens.

CMT-2	Uninformative names of an event and a function
File	PoolingStakingContractV2.sol
Severity	Comment
Status	Fixed at 90b2c476

DESCRIPTION

At the lines:

- PoolingStakingContractV2.sol#L28
- PoolingStakingContractV2.sol#L152

RECOMMENDATION

Make the names more informative (E.g. `DepositIsEnabledSet` and `getAvailableDelegatorReward`[to avoid double naming])

CMT-3	Rough BASIS_FRACTION
File	PoolingStakingContractV2.sol
Severity	Comment
Status	Fixed at 90b2c476

DESCRIPTION

At the line PoolingStakingContractV2.sol#L37
`BASIS_FRACTION = 100` is too rough.

RECOMMENDATION

It would be better to set it to 10000 or even more to increase accuracy.

CMT-4	Const `workerOwner`
File	PoolingStakingContractV2.sol
Severity	Comment
Status	No issue

DESCRIPTION

PoolingStakingContractV2.sol#L40
It's not clear why `workerOwner` is a `const`.

RECOMMENDATION

If it's not supposed to be transferred, write a comment describing this.

CMT-5	Explicit visibility of `workerFraction`
File	PoolingStakingContractV2.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

Explicit statements make code more readable.
PoolingStakingContractV2.sol#L46

RECOMMENDATION

Add `public` visibility modifier

CLIENT'S COMMENTARY

> Contract is using also as demonstration/example, so `getWorkerFraction` is the main place to calculate final value for worker fraction

CMT-6	External `initialize`
File	PoolingStakingContractV2.sol
Severity	Comment
Status	Fixed at 90b2c476

DESCRIPTION

Since it will be called by the end-user only, it's better to use `external` modifier: PoolingStakingContractV2.sol#L59

RECOMMENDATION

Use `external` modifier

CMT-7	`getWorkerFraction` is not needed
File	PoolingStakingContractV2.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

Public attributes already have getters
PoolingStakingContractV2.sol#L91

RECOMMENDATION

Remove the method `getWorkerFraction`.

CMT-8	Comment required
File	PoolingStakingContractV2.sol
Severity	Comment
Status	Fixed at 90b2c476

DESCRIPTION

It's not intuitive what is happening here

- PoolingStakingContractV2.sol#L116
- PoolingStakingContractV2.sol#L140
-

RECOMMENDATION

Add the comment to the code.

CMT-9	Avoid ternary
File	PoolingStakingContractV2.sol
Severity	Comment
Status	Fixed at 90b2c476

DESCRIPTION

If-else statement is more clear way for complex conditions

- PoolingStakingContractV2.sol#L168

RECOMMENDATION

Use if-else statement.

CMT-10	Not clear TODO-comment
File	PoolingStakingContractV2.sol
Severity	Comment
Status	Fixed at 90b2c476

DESCRIPTION

It's not clear what is that `TODO` about
PoolingStakingContractV2.sol#L226

RECOMMENDATION

Add more details into commentary

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