Alium Finance Smart Contract final Audit Report

Overview

Scope of Audit

The scope of this audit was to analyse and document the **Alium Finance** smart contract codebase for quality, security, and correctness.

Check Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked maths
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contracts, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods, and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational Issues

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of security issues per severity.

TYPE	HIGH	MEDIUM	LOW	INFORMATIONAL
Open	0	0	1	1
Acknowledged	0	1	0	0
Closed	0	1	3	1

Introduction

During the period of **Feb 16, 2022 to Feb 21, 2022** - QuillAudits Team performed a security audit for **Alium Finance** smart contracts.

The code for the audit was taken from following the official link**:**

Codebase: d3ee2babf96243b262d60c267e0ad1056152d2d9

Issues Found – Code Review / Manual Testing

High Severity Issues

none

Medium Severity Issues

- **[M1] reentrancy in staking and withdraw function**

  In stake/withdraw function the transferFrom is called before the state update and the reentrancy is seen in the function.

  We recommend to use CheckEffectsInteractions pattern throughout the contract or use Openzepplin reentrancy guard

  Status: **Closed**

- **[M2] Weak PRNG.**

  The pseudo-random number generator (**PRNG**) is weak and inefficient.

  Everything in blockchain is visible and publicly available. So it's always advised not to generate random numbers in the contract.

  We recommend using oracle (outside random data source) for generation of randomness in the contract instead of making our own random function.

  Status: **Acknowledged**

Low Severity Issues

- **[L1] calls inside loop**

  The transferFrom call in withdraw function is inside for loop which can lead to DOS (denial of service attack) and waste of gas.

We recommend it to use the transferFrom outside the for loop and transfer all amount in only one transaction.

Reference - #calls-inside-a-loop

Status: Open

- **[L2] Used locked pragma version:**

  The pragma versions used in the contract are not locked. Consider using the latest versions among 0.8.11 for deploying the contracts and libraries as it does not compile for any other version and can be confusing for a developer. Solidity source files indicate the versions of the compiler they can be compiled with.

  ```
  pragma solidity ^0.8.0; // bad: compiles between 0.8.0 and 0.8.11
  pragma solidity 0.8.0; // good : compiles w 0.8.0 only but not the latest version
  pragma solidity 0.8.11; // best: compiles w 0.8.11
  ```

  Status: Closed

- **[L3] Unused variable:**

  The variable ***uint256 num*** is not being used anywhere in the contract we suggest to remove from the contract and save deployment gas price.

  Status: Closed

- **[L4] transferFrom return value is ignored.**

  The transferFrom return value is ignored.

  We recommend to use `SafeERC20`, or ensure that the transfer/transferFrom return value is checked

  Status: Closed

Informational Issues

- **[INF1] Missing comments and description:**

  Comments and Description of the methods and the variables are missing, it's hard to read and understand the purpose of the variables and the methods in context of the whole picture

  Recommendation: Consider adding NatSpec format comments for the comments and state variables

  Status: Open

- **[INF2] Public methods only being used externally**

  'public' functions that are never used within the contract should be declared 'external' to save gas.

  Recommendation: Make these methods external

  saveCellPrice, saveCellBonusChance, saveCellDuration, saveCellRewards, saveCellRandomBonuses, saveMonsterImageUrl, saveMonsterName, saveRegionName, addCell, enableCell, disableCell, myStakeInfo, pause, unpause, stake, withdraw, editStakeToken, editStakeTokenAdmin, getUsers, .

  Status: Closed

Mumbai Testnet Test Contract

Contract : 0x594714b143FD58c481c1Af95Fb1FDA5704DC2176

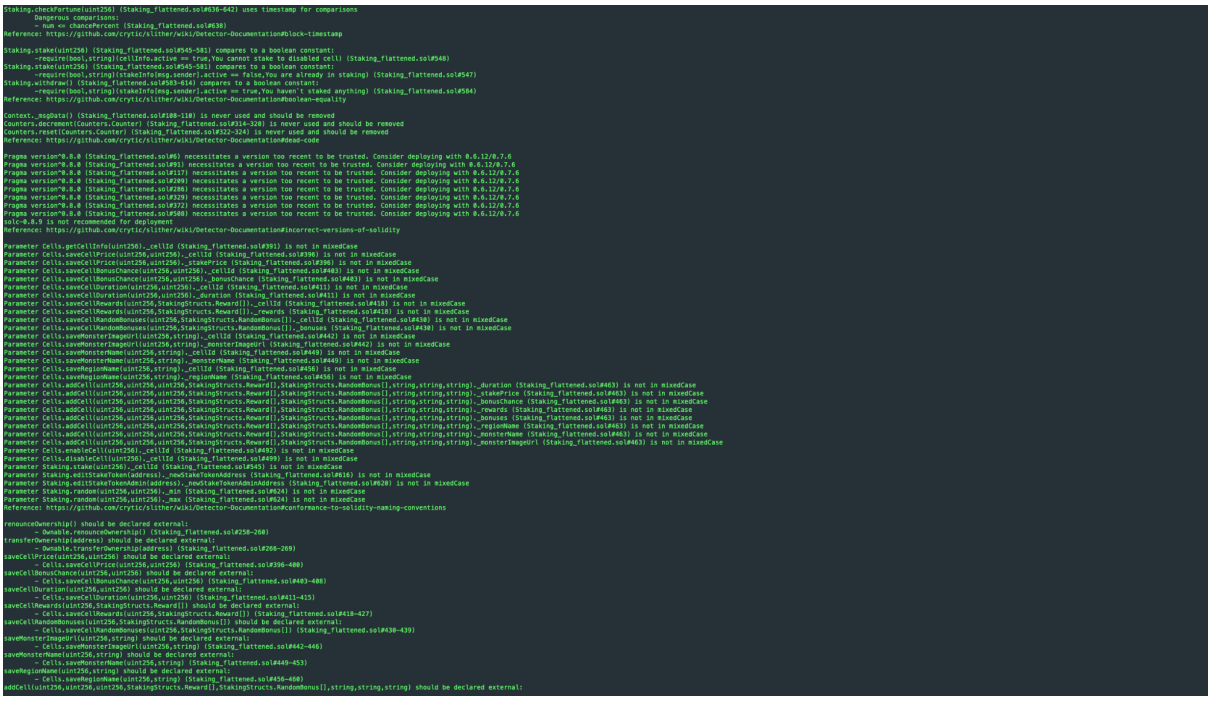
Functional Tests

- Add new Cell - PASS
- Only owner can add new Cell - PASS
- No approval for staking contract - PASS
- Approve stake price to staking contract - PASS
- Stake cell ID-1 - PASS
- Already stake for cell ID-1 - PASS
- Withdraw no stake tokens - PASS
- Withdraw amount is more than allowance - PASS
- Approve 1500 to staking contract to send bonus/reward to user - PASS
- Withdraw successfully - PASS
- Owner add new cellID-2 - PASS
- Approve more than stake price to staking contract - PASS
- User2 stake cellI D-2 - PASS
- Withdraw before the staking time end - PASS
- Withdraw user stake and send bonus/reward to user - PASS

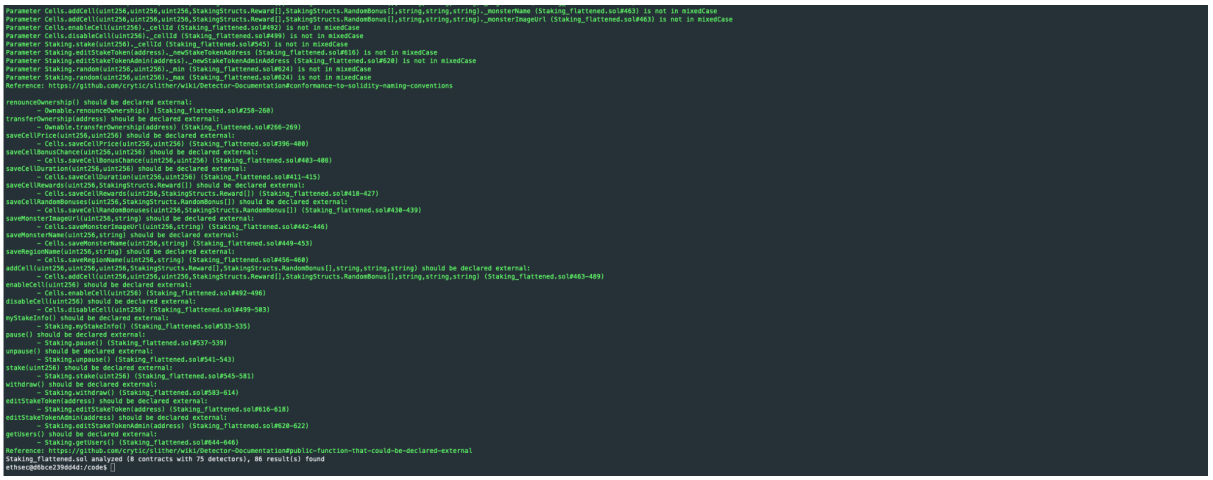
Automated Tests

Slither:

```
Staking.random(uint256,uint256) (Staking_flattened.sol#624-634) uses a weak PRNG: "seed = uint256(keccak256(bytes)(abi.encodePacked(block.timestamp + block.difficulty + ((uint256(keccak256(bytes)(abi.encodePacked(block.coinbase)))) / (block.timestamp)) + block.gaslimit + ((uint256(keccak256(bytes)(abi.encodePacked(msg.sender)))) / (block.timestamp)) + block.number))) % diff (Staking_flattened.sol#624-632)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Staking.stake(uint256) (Staking_flattened.sol#545-581) ignores return value by stakeToken.transferFrom(msg.sender,stakeTokenAdminAddress,cellInfo.stakePrice) (Staking_flattened.sol#549)
Staking.withdraw() (Staking_flattened.sol#583-614) ignores return value by rewardInstance.transferFrom(stakeTokenAdminAddress,msg.sender,rewardAmount) (Staking_flattened.sol#591)
Staking.withdraw() (Staking_flattened.sol#583-614) ignores return value by bonusInstance.transferFrom(stakeTokenAdminAddress,msg.sender,bonusAmount) (Staking_flattened.sol#606)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

Reentrancy in Staking.stake(uint256) (Staking_flattened.sol#545-581):
        External calls:
        - stakeToken.transferFrom(msg.sender,stakeTokenAdminAddress,cellInfo.stakePrice) (Staking_flattened.sol#549)
        State variables written after the call(s):
        - stakeInfo[msg.sender].active = true (Staking_flattened.sol#550)
        - stakeInfo[msg.sender].cellId = _cellId (Staking_flattened.sol#551)
        - stakeInfo[msg.sender].bonusChance = cellInfo.bonusChance (Staking_flattened.sol#552)
        - stakeInfo[msg.sender].endTimestamp = block.timestamp + cellInfo.stakeDuration (Staking_flattened.sol#553)
        - stakeInfo[msg.sender].regionName = cellInfo.regionName (Staking_flattened.sol#554)
        - stakeInfo[msg.sender].monsterName = cellInfo.monsterName (Staking_flattened.sol#555)
        - stakeInfo[msg.sender].monsterImageUrl = cellInfo.monsterImageUrl (Staking_flattened.sol#556)
        - delete stakeInfo[msg.sender].rewards (Staking_flattened.sol#558)
        - rewardInd < cellInfo.rewards.length (Staking_flattened.sol#560)
        - stakeInfo[msg.sender].rewards.push(oneReward) (Staking_flattened.sol#563)
        - delete stakeInfo[msg.sender].bonuses (Staking_flattened.sol#566)
        - bonusInd < cellInfo.bonuses.length (Staking_flattened.sol#568)
        - stakeInfo[msg.sender].bonuses.push(oneBonus) (Staking_flattened.sol#572)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Staking.stake(uint256).cellInfo (Staking_flattened.sol#546) shadows:
        - Cells.cellInfo (Staking_flattened.sol#382) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Staking.constructor(address,address)._stakeTokenAdminAddress (Staking_flattened.sol#528) lacks a zero-check on :
        - stakeTokenAdminAddress = _stakeTokenAdminAddress (Staking_flattened.sol#530)
Staking.editStakeTokenAdmin(address)._newStakeTokenAdminAddress (Staking_flattened.sol#620) lacks a zero-check on :
        - stakeTokenAdminAddress = _newStakeTokenAdminAddress (Staking_flattened.sol#621)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Staking.withdraw() (Staking_flattened.sol#583-614) has external calls inside a loop: rewardInstance.transferFrom(stakeTokenAdminAddress,msg.sender,rewardAmount) (Staking_flattened.sol#591)
Staking.withdraw() (Staking_flattened.sol#583-614) has external calls inside a loop: bonusInstance.transferFrom(stakeTokenAdminAddress,msg.sender,bonusAmount) (Staking_flattened.sol#606)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#calls-inside-a-loop

Reentrancy in Staking.stake(uint256) (Staking_flattened.sol#545-581):
        External calls:
        - stakeToken.transferFrom(msg.sender,stakeTokenAdminAddress,cellInfo.stakePrice) (Staking_flattened.sol#549)
        State variables written after the call(s):
        - activeStakesCount += (Staking_flattened.sol#579)
        - usersExist[msg.sender] = true (Staking_flattened.sol#578)
        - usersList.push(msg.sender) (Staking_flattened.sol#576)
Reentrancy in Staking.withdraw() (Staking_flattened.sol#583-614):
        External calls:
        - rewardInstance.transferFrom(stakeTokenAdminAddress,msg.sender,rewardAmount) (Staking_flattened.sol#591)
        State variables written after the call(s):
        - statRewards[rewardAddress] += rewardAmount (Staking_flattened.sol#592)
Reentrancy in Staking.withdraw() (Staking_flattened.sol#583-614):
        External calls:
        - bonusInstance.transferFrom(stakeTokenAdminAddress,msg.sender,bonusAmount) (Staking_flattened.sol#606)
        State variables written after the call(s):
        - statRewards[bonusAddress] += bonusAmount (Staking_flattened.sol#607)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Staking.stake(uint256) (Staking_flattened.sol#545-581):
        External calls:
        - stakeToken.transferFrom(msg.sender,stakeTokenAdminAddress,cellInfo.stakePrice) (Staking_flattened.sol#549)
        Event emitted after the call(s):
        - Stake(msg.sender,_cellId,stakeInfo[msg.sender].endTimestamp) (Staking_flattened.sol#580)
Reentrancy in Staking.withdraw() (Staking_flattened.sol#583-614):
        External calls:
        - rewardInstance.transferFrom(stakeTokenAdminAddress,msg.sender,rewardAmount) (Staking_flattened.sol#591)
        Event emitted after the call(s):
        - Reward(msg.sender,rewardAddress,rewardAmount) (Staking_flattened.sol#593)
Reentrancy in Staking.withdraw() (Staking_flattened.sol#583-614):
        External calls:
        - bonusInstance.transferFrom(stakeTokenAdminAddress,msg.sender,bonusAmount) (Staking_flattened.sol#606)
        Event emitted after the call(s):
        - Reward(msg.sender,bonusAddress,bonusAmount) (Staking_flattened.sol#608)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Staking.withdraw() (Staking_flattened.sol#583-614) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(stakeInfo[msg.sender].endTimestamp < block.timestamp,Staking time not ended) (Staking_flattened.sol#585)
Staking.checkFortune(uint256) (Staking_flattened.sol#636-642) uses timestamp for comparisons
```



```
Staking.checkFortune(uint256) (Staking_flattened.sol#636-642) uses timestamp for comparisons
        Dangerous comparisons:
        - num <= chancePercent (Staking_flattened.sol#638)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Staking.stake(uint256) (Staking_flattened.sol#545-581) compares to a boolean constant:
        -require(bool,string)(cellInfo.active == true,You cannot stake to disabled cell) (Staking_flattened.sol#548)
Staking.stake(uint256) (Staking_flattened.sol#545-581) compares to a boolean constant:
        -require(bool,string)(stakeInfo[msg.sender].active == false,You are already in staking) (Staking_flattened.sol#547)
Staking.withdraw() (Staking_flattened.sol#583-614) compares to a boolean constant:
        -require(bool,string)(stakeInfo[msg.sender].active == true,You haven't staked anything) (Staking_flattened.sol#584)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Context._msgData() (Staking_flattened.sol#108-110) is never used and should be removed
Counters.decrement(Counters.Counter) (Staking_flattened.sol#314-320) is never used and should be removed
Counters.reset(Counters.Counter) (Staking_flattened.sol#322-324) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (Staking_flattened.sol#6) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Staking_flattened.sol#91) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Staking_flattened.sol#117) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Staking_flattened.sol#299) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Staking_flattened.sol#286) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Staking_flattened.sol#329) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Staking_flattened.sol#372) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Staking_flattened.sol#508) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter Cells.getCellInfo(uint256)._cellId (Staking_flattened.sol#391) is not in mixedCase
Parameter Cells.saveCellPrice(uint256,uint256)._cellId (Staking_flattened.sol#396) is not in mixedCase
Parameter Cells.saveCellPrice(uint256,uint256)._stakePrice (Staking_flattened.sol#396) is not in mixedCase
Parameter Cells.saveCellBonusChance(uint256,uint256)._cellId (Staking_flattened.sol#403) is not in mixedCase
Parameter Cells.saveCellBonusChance(uint256,uint256)._bonusChance (Staking_flattened.sol#403) is not in mixedCase
Parameter Cells.saveCellDuration(uint256,uint256)._cellId (Staking_flattened.sol#411) is not in mixedCase
Parameter Cells.saveCellDuration(uint256,uint256)._duration (Staking_flattened.sol#411) is not in mixedCase
Parameter Cells.saveCellRewards(uint256,StakingStructs.Reward[])._cellId (Staking_flattened.sol#418) is not in mixedCase
Parameter Cells.saveCellRewards(uint256,StakingStructs.Reward[])._rewards (Staking_flattened.sol#418) is not in mixedCase
Parameter Cells.saveCellRandomBonuses(uint256,StakingStructs.RandomBonus[])._cellId (Staking_flattened.sol#430) is not in mixedCase
Parameter Cells.saveCellRandomBonuses(uint256,StakingStructs.RandomBonus[])._bonuses (Staking_flattened.sol#430) is not in mixedCase
Parameter Cells.saveMonsterImageUrl(uint256,string)._cellId (Staking_flattened.sol#442) is not in mixedCase
Parameter Cells.saveMonsterImageUrl(uint256,string)._monsterImageUrl (Staking_flattened.sol#442) is not in mixedCase
Parameter Cells.saveMonsterName(uint256,string)._cellId (Staking_flattened.sol#449) is not in mixedCase
Parameter Cells.saveMonsterName(uint256,string)._monsterName (Staking_flattened.sol#449) is not in mixedCase
Parameter Cells.saveRegionName(uint256,string)._cellId (Staking_flattened.sol#456) is not in mixedCase
Parameter Cells.saveRegionName(uint256,string)._regionName (Staking_flattened.sol#456) is not in mixedCase
Parameter Cells.addCell(uint256,uint256,uint256,StakingStructs.Reward[],StakingStructs.RandomBonus[],string,string,string)._duration (Staking_flattened.sol#463) is not in mixedCase
Parameter Cells.addCell(uint256,uint256,uint256,StakingStructs.Reward[],StakingStructs.RandomBonus[],string,string,string)._stakePrice (Staking_flattened.sol#463) is not in mixedCase
Parameter Cells.addCell(uint256,uint256,uint256,StakingStructs.Reward[],StakingStructs.RandomBonus[],string,string,string)._bonusChance (Staking_flattened.sol#463) is not in mixedCase
Parameter Cells.addCell(uint256,uint256,uint256,StakingStructs.Reward[],StakingStructs.RandomBonus[],string,string,string)._rewards (Staking_flattened.sol#463) is not in mixedCase
Parameter Cells.addCell(uint256,uint256,uint256,StakingStructs.Reward[],StakingStructs.RandomBonus[],string,string,string)._bonuses (Staking_flattened.sol#463) is not in mixedCase
Parameter Cells.addCell(uint256,uint256,uint256,StakingStructs.Reward[],StakingStructs.RandomBonus[],string,string,string)._regionName (Staking_flattened.sol#463) is not in mixedCase
Parameter Cells.addCell(uint256,uint256,uint256,StakingStructs.Reward[],StakingStructs.RandomBonus[],string,string,string)._monsterName (Staking_flattened.sol#463) is not in mixedCase
Parameter Cells.addCell(uint256,uint256,uint256,StakingStructs.Reward[],StakingStructs.RandomBonus[],string,string,string)._monsterImageUrl (Staking_flattened.sol#463) is not in mixedCase
Parameter Cells.enableCell(uint256)._cellId (Staking_flattened.sol#492) is not in mixedCase
Parameter Cells.disableCell(uint256)._cellId (Staking_flattened.sol#499) is not in mixedCase
Parameter Staking.stake(uint256)._cellId (Staking_flattened.sol#545) is not in mixedCase
Parameter Staking.editStakeTokenAdmin(address)._newStakeTokenAdminAddress (Staking_flattened.sol#616) is not in mixedCase
Parameter Staking.editStakeTokenAdmin(address)._newStakeTokenAdminAddress (Staking_flattened.sol#620) is not in mixedCase
Parameter Staking.random(uint256,uint256)._min (Staking_flattened.sol#624) is not in mixedCase
Parameter Staking.random(uint256,uint256)._max (Staking_flattened.sol#624) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Staking_flattened.sol#258-260)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Staking_flattened.sol#266-269)
saveCellPrice(uint256,uint256) should be declared external:
        - Cells.saveCellPrice(uint256,uint256) (Staking_flattened.sol#396-400)
saveCellBonusChance(uint256,uint256) should be declared external:
        - Cells.saveCellBonusChance(uint256,uint256) (Staking_flattened.sol#403-408)
saveCellDuration(uint256,uint256) should be declared external:
        - Cells.saveCellDuration(uint256,uint256) (Staking_flattened.sol#411-415)
saveCellRewards(uint256,StakingStructs.Reward[]) should be declared external:
        - Cells.saveCellRewards(uint256,StakingStructs.Reward[]) (Staking_flattened.sol#418-427)
saveCellRandomBonuses(uint256,StakingStructs.RandomBonus[]) should be declared external:
        - Cells.saveCellRandomBonuses(uint256,StakingStructs.RandomBonus[]) (Staking_flattened.sol#430-439)
saveMonsterImageUrl(uint256,string) should be declared external:
        - Cells.saveMonsterImageUrl(uint256,string) (Staking_flattened.sol#442-446)
saveMonsterName(uint256,string) should be declared external:
        - Cells.saveMonsterName(uint256,string) (Staking_flattened.sol#449-453)
saveRegionName(uint256,string) should be declared external:
        - Cells.saveRegionName(uint256,string) (Staking_flattened.sol#456-460)
addCell(uint256,uint256,uint256,StakingStructs.Reward[],StakingStructs.RandomBonus[],string,string,string) should be declared external:
```



Results:

No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorised above according to their level of severity.

Closing Summary

The listed issues of Medium, Low severity and information issues are fixed/acknowledged.
The code is good to release on the public mainnet.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or endorsement of the **Alium Finance platform**. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the **Alium Finance** Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.