UPLIFT

SMART CONTRACT AUDIT

ZOKYO.

Dec 31st, 2021 | v. 1.0

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

TECHNICAL SUMMARY

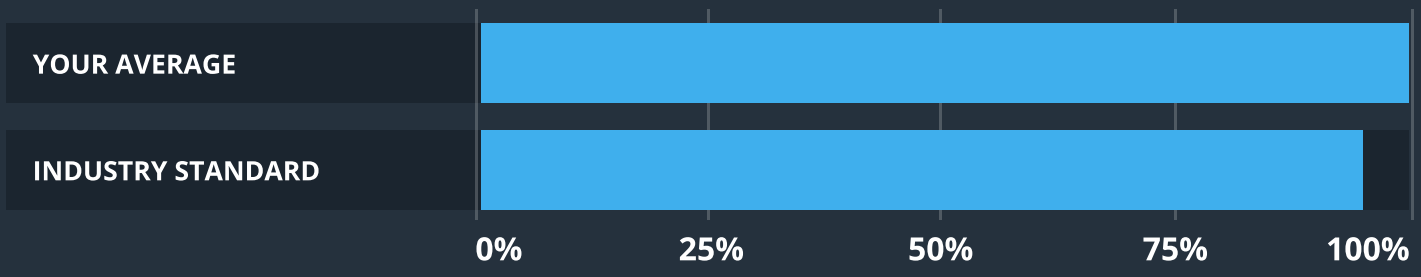
This document outlines the overall security of the Uplift smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the Uplift smart contract codebase for quality, security, and correctness.

Contract Status

There was 1 critical issue found during the audit.

Testable Code



The testable code is 99.81%, which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Zokyo recommend that the Uplift team put in place a bug bounty program to encourage further and active analysis of the smart contract.

TABLE OF CONTENTS

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the Uplift archive hash.

Archive hash:
0c645485fc66100f8eb75f4332661b62babec91a831af204c72a6e007e280dd8

Last commit:
https://github.com/mariabeyrak/staking-uplift/commit/6817b52d56efb2a4976bfacbbde802708abcb487

Contracts:

- Staking;
- ThrottledPool;
- StakingPool;
- IdPool;
- Pool.

Throughout the review process, care was taken to ensure that the token contract:

- Implements and adheres to existing Token standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of gas, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of Uplift smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1	Due diligence in assessing the overall code quality of the codebase.	3	Testing contract logic against common and uncommon attack vectors.
2	Cross-comparison with other, similar smart contracts by industry leaders.	4	Thorough, manual review of the codebase, line-by-line.

SUMMARY

The Zokyo team has conducted a security audit of the given codebase. All the findings within the auditing process are presented in this document.

There was 1 critical issue found during the auditing process. 1 high, 2 medium, and 2 informational issues were found. All of the issues found have been resolved by the Uplift team. During the audit, progress was changed to a logic of contacts and some issues are not relevant.

Hence, the score of the audit is set to 98 to the provided codebase.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the ability of the contract to compile or operate in a significant way.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

Method _stake of contract Staking has Incorrect nextTierIndex calculation

CRITICAL | NOT VALID

Snippet:

```
uint8 nextTierIndex = _row * (rowsCount + 1) + _column;
```

Recommendation:
For the last two tiers, nextTierIndex will exceed the array length of tierSnapshots_.

State variable minReferrerStakeAmount could be defined by contract owner and influence being as referral

HIGH | NOT VALID

Recommendation:
Add details on minimal staking amount to be qualified as referral.

Solidity compiler pragma is not pointing to the exact version

MEDIUM | UNRESOLVED

Recommendation:
Define specific Solidity compiler version in pragma keyword.

Method canParticipate of contract Staking is not used

MEDIUM | RESOLVED

It is a bad approach to include methods that are not utilized by Smart contracts, as they increase deployed byte code.

Recommendation:
Remove unnecessary methods from the smart contract.

Comment:
It is used for further development.

Staking contract could be upgraded by the contract owner

INFORMATIONAL | UNRESOLVED

The contract owner has a right to upgrade this smart contract. Hence, that may lead to potential vulnerability after SC upgrade so that there is room for having potential operational or security issues.

Recommendation:
Remove the permission to upgrade the contract by the contract owner.

Smart contract is not covered by NatSpec annotations

INFORMATIONAL | RESOLVED

Recommendation:
Cover by NatSpec all Contract methods.

	Staking	Pool	IdPool
Re-entrancy	Pass	Pass	Pass
Access Management Hierarchy	Pass	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass	Pass
Unexpected Ether	Pass	Pass	Pass
Delegatecall	Pass	Pass	Pass
Default Public Visibility	Pass	Pass	Pass
Hidden Malicious Code	Pass	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass	Pass
External Contract Referencing	Pass	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass	Pass
Unchecked CALL Return Values	Pass	Pass	Pass
Race Conditions / Front Running	Pass	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass	Pass
Floating Points and Precision	Pass	Pass	Pass
Tx.Origin Authentication	Pass	Pass	Pass
Signatures Replay	Pass	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass	Pass

	ThrottledPool	StakingPool
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests are written by the Uplift team

Code Coverage

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts\	94.37	87.50	90.00	94.66	
BaseProxy.sol	100.00	100.00	100.00	100.00	
IdPool.sol	100.00	79.17	100.00	100.00	
LIFT.sol	100.00	100.00	100.00	100.00	
MerkleTreeWhitelist.sol	59.09	50.00	50.00	59.09	... 46,47,56,74
Pool.sol	100.00	90.91	100.00	100.00	
ReferrersData.sol	73.68	60.00	57.14	75.00	24,25,29,30,31
Registry.sol	100.00	100.00	100.00	100.00	
Staking.sol	97.22	91.46	96.97	97.25	... 257,373,578
StakingPool.sol	100.00	100.00	100.00	100.00	
ThrottledPool.sol	94.44	93.75	83.33	95.24	34,35
All files	**94.37**	**87.50**	**90.00**	**94.66**	

Test Results

Contract: IdPool

- ✓ mintForId:restricted (38ms)
- ✓ burnForId:restricted
- ✓ mint:withdraw:2:one account (1414ms)
- ✓ mint:withdraw:2:two accounts (2234ms)

LIFT

- ✓ name, symbol, decimals
- ✓ INITIAL_SUPPLY
- ✓ MAX_SUPPLY
- ✓ mint
- ✓ burn
- ✓ grantRole, revokeRole (207ms)
- ✓ changeOwner (148ms)
- ✓ permit (192ms)

Pool

- ✓ setToken:restricted
- ✓ setToken:invalid
- ✓ setToken:success
- ✓ balanceOf, totalBalanceOf, totalSupply, withdraw, withdrawableRewardsOf (2345ms)
- ✓ mint:withdraw:2:one account (703ms)

Staking

- ✓ initialize (1634ms)
- ✓ setMinTierReferrerBooster:restricted
- ✓ setMinTierReferrerBooster:invalid
- ✓ setMinTierReferrerBooster (88ms)
- ✓ stake:invalid amount (133ms)
- ✓ stake:zero
- ✓ stake:no time:less than 100 (2582ms)
- ✓ stake:no time:exact amount (2584ms)
- ✓ stake:no time:between two tiers (2849ms)
- ✓ stake:unstake:no ref (2470ms)
- ✓ stake:unstake:linear:less than first tier (4680ms)
- ✓ stake:unstake:between two tiers (2633ms)
- ✓ stake:unstake:linear:between two tiers (4601ms)
- ✓ stake:unstake:linear:between two last tiers (893ms)
- ✓ stake:unstake:linear:last tier (4539ms)

```
    ✓ stake:unstake:linear:more than last tier (4597ms)
    ✓ stake:invalid params (38ms)
    ✓ stake user:stake referrer:stake user 2:unstake user1 (8439ms)
    ✓ stake:unstake:invalid id (843ms)
    ✓ stake:unstake:early exit (2768ms)
    1) stake:unstake:early exit:exact tier:no ref
    2)  stake:unstake:early exit:exact tier:ref
    ✓ stake:unstake:early exit:less than first tier (2722ms)
    3) stake:unstake:early exit:between two tiers
    ✓ stake:unstake:early exit:more than last tier (2774ms)
    ✓ stake:self referring (54ms)
    ✓ stake:circle referring 910ms)
    ✓ stake:invalid referrer (1770ms)
    ✓ stakeWithPermit (885ms)
    4) stake:max referrer booster
    ✓ stake1:stake2:unstake1 (3382ms)
    changeTiers
      ✓ changeTiers:restricted
      ✓ changeTiers:invalid arguments
      ✓ changeTiers (1402ms)
      ✓ stake:changeTiers:stake:unstake first (4529ms)
    stakingPower
      ✓ initialize
      ✓ set:stakingPowerInitialBreak:participationBreak (103ms)
      ✓ stakingPower flow (2499ms)
      ✓ updateStakingPower:10 times:setParticipationBreak:burnStakingPower (10930ms)
    kyc
      ✓ setWhitelist (87ms)
      ✓ setMinReferrerStakeAmount (88ms)
      ✓ kyc (3328ms)
  Contract: StakingPool
    ✓ mintForId:burnForId:restricted
    ✓ mint:burn:mintForId:withdraw:withdrawForId:restricted (439ms)
    ✓ mint:withdrawForAccount:restricted (542ms)
    ✓ mint:withdrawForAccount:100 percent (1108ms)
    ✓ mint:withdrawForAccount:50 percent (1597ms)
    ✓ mint:withdrawForAccount:invalid amount (480ms)
    ✓ mint:withdrawForAccount:fee (1199ms)
    ✓ mint:withdrawForAccount:invalid fee (473ms)
```

StakingUpgradeability
✓ upgradeability (168ms)
✓ upgradeability:owner

ThrottledPool
✓ setEmissionController:restricted
✓ setEmissionController:invalid
✓ setEmissionController:success
✓ mint:withdrawableRewardsOf:withdraw:mint2:withdraw2:less than total (2530ms)
✓ mint:withdrawableRewardsOf:withdraw:more than total (1465ms)

70 passing (3m)
4 failing

1) Staking
stake:unstake:early exit:exact tier:no ref:

AssertionError: Expected "1289682143564356435643562" to be equal
128968114851485148514851 2
+ expected - actual

{
- "_hex": "0x011119c1519eaeb3486560"
+ "_hex": "0x011119cf20beef67c90caa"
"_isBigNumber": true
}

at assertArgsArraysEqual (node_modules/@ethereum-waffle/chai/dist/cjs/matchers/emit.js:58:54)
at tryAssertArgsArraysEqual (node_modules/@ethereum-waffle/chai/dist/cjs/matchers/emit.js:65:20)
at /home/romario/Work/Audit/new-uplift/staking-uplift/node_modules/@ethereum-waffle/chai/dist/cjs/matchers/emit.js:77:13
at runMicrotasks (<anonymous>)
at processTicksAndRejections (internal/process/task_queues.js:97:5)
at runNextTicks (internal/process/task_queues.js:66:3)
at listOnTimeout (internal/timers.js:523:9)
at processTimers (internal/timers.js:497:7)

2) Staking
stake:unstake:early exit:exact tier:ref:

```
    AssertionError: Expected "1289682143564356435643562" to be equal
128968114851485148514851 2
    + expected - actual

     {
    -  "_hex": "0x011119c1519eaeb3486560"
    +  "_hex": "0x011119cf20beef67c90caa"
       "_isBigNumber": true
     }

    at assertArgsArraysEqual (node_modules/@ethereum-waffle/chai/dist/cjs/matchers/emit.js:58:54)
    at tryAssertArgsArraysEqual (node_modules/@ethereum-waffle/chai/dist/cjs/matchers/
emit.js:65:20)
    at /home/romario/Work/Audit/new-uplift/staking-uplift/node_modules/@ethereum-waffle/chai/
dist/cjs/matchers/emit.js:77:13
    at runMicrotasks (<anonymous>)
    at processTicksAndRejections (internal/process/task_queues.js:97:5)
    at runNextTicks (internal/process/task_queues.js:66:3)
    at listOnTimeout (internal/timers.js:523:9)
    at processTimers (internal/timers.js:497:7)

3) Staking
     stake:unstake:early exit:between two tiers:

    AssertionError: Expected "2069751970815724815724815 5" to be equal
2069751871002457002457002 2
    + expected - actual

     {
    -  "_hex": "0x111edebcce556610152ca6"
    +  "_hex": "0x111edecaa869c7f6e3ea9b"
       "_isBigNumber": true
     }

    at assertArgsArraysEqual (node_modules/@ethereum-waffle/chai/dist/cjs/matchers/emit.js:58:54)
at tryAssertArgsArraysEqual (node_modules/@ethereum-waffle/chai/dist/cjs/matchers/emit.js:65:20)
    at /home/romario/Work/Audit/new-uplift/staking-uplift/node_modules/@ethereum-waffle/chai/
dist/cjs/matchers/emit.js:77:13
    at runMicrotasks (<anonymous>)
```

```
      at processTicksAndRejections (internal/process/task_queues.js:97:5)
      at runNextTicks (internal/process/task_queues.js:66:3)
      at listOnTimeout (internal/timers.js:523:9)
      at processTimers (internal/timers.js:497:7)

4) Staking
       stake:max referrer booster:
     Error: Timeout of 20000ms exceeded. For async tests and hooks, ensure "done()" is called; if
returning a Promise, ensure it resolves. (/home/romario/Work/Audit/new-uplift/staking-uplift/test/
Staking.ts)
      at listOnTimeout (internal/timers.js:554:17)
      at processTimers (internal/timers.js:497:7)
```

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Security team

As part of our work assisting Uplift in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Truffle testing framework.

Tests were based on the functionality of the code, as well as a review of the Uplift contract requirements for details about issuance amounts and how the system handles these.

Code Coverage

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
IdPool.sol	100.00	91.67	100.00	100.00	
Pool.sol	100.00	95.45	100.00	100.00	
Registry.sol	100.00	100.00	100.00	100.00	
Staking.sol	98.88	93.75	100.00	98.90	378,521
StakingPool.sol	100.00	100.00	100.00	100.00	
ThrottledPool.sol	94.44	100.00	100.00	100.00	34,35
All files	**99.81**	**96.81**	**100.00**	**99.42**	

Test Results

```
Contract: IdPool
  Check constructor
    ✓ should revert if registry zero address (47ms)
    ✓ should revert if token zero address
    ✓ should set registry address
    ✓ should set token address
  Check constructor
    mintForId
      ✓ should revert if not the owner
      ✓ should increase balance if _totalAmount > 0 (736ms)
      ✓ should increase totalSupply if _totalAmount > 0 (670ms)
      ✓ should increase idBalanceOf if _idAmount > 0 (562ms)
      ✓ should increase accountTotalSupply if _idAmount > 0 (487ms)
      ✓ should catch event Mint, and MintForId (244ms)
    check withdrawableRewardsOf & withdrawableRewardsForId
      ✓ should calculate correct amount for both users (794ms)
   burnForId
    ✓ should revert if not the owner
    ✓ should decrease balance if _totalAmount > 0 (274ms)
    ✓ should decrease totalSupply if _totalAmount > 0 (197ms)
    ✓ should decrease idBalanceOf if _idAmount > 0 (151ms)
    ✓ should decrease accountTotalSupply if _idAmount > 0 (183ms)
    ✓ should catch event BurnForId (142ms)
    check withdrawableRewardsOf & withdrawableRewardsForId
      ✓ should calculate correct amount for both users (691ms) (691ms)
   withdrawForId
    1) should update { tokenRewardsOf[account] & idTokenRewardsOf[account][_id] }
    ✓ should transfer reward (437ms)
    ✓ should catch event WithdrawForId (155ms)
  withdrawableRewardsForId
    2) should return correct withdrawable rewards

  IdPool
    Check constructor
    ✓ should revert if registry zero address
    ✓ should revert if token zero address
```

```
✓ should set registry address
✓ should set token address
Check functions:
  setToken
    ✓ should revert if zero address
    ✓ should set token address
    ✓ should catch event SetToken
  mint
    ✓ should revert if not the owner
    ✓ should increase balance (248ms)
    ✓ should increase totalSupply (245ms)
    ✓ should catch event Mint (64ms)
    check withdrawableRewardsOf
      ✓ should calculate correct amount for both users (427ms)
  burn
    ✓ should revert if not the owner
    ✓ should decrease balance (391ms)
    ✓ should decrease totalSupply (204ms)
    ✓ should catch event Burn (103ms)
    check withdrawableRewardsOf
      ✓ should calculate correct amount for both users (372ms)
  withdraw
    ✓ should calculate reward correctly (786ms)
    ✓ should update { tokenRewardsOf[account] } (224ms)
    ✓ should transfer reward (218ms)
    ✓ should catch event Withdraw (104ms)
  withdrawableRewardsForId
    ✓ should return correct withdrawable rewards (678ms)

Staking
  Check functions:
    initialize
    ✓ should initialize correctly
    ✓ should revert if contract already initialized
    ✓ should revert if address of parameters is zero's address
    ✓ should revert if parameters of tier is invalid
    ✓ should revert if previous tier more than next tier
    setWhitelist
    ✓  should set whitelist correctly (80ms)
```

✓ should revert if caller haven't role of admin

setStakingPowerData

✓ should set staking power data correctly (91ms)

✓ should set staking power data correctly (83ms)

✓ should revert if caller haven't role of admin

setMinTierReferrerBooster

✓ should set min tier referrer booster correctly (80ms)

✓ should revert if new value more than length of tier (248ms)

setMinReferrerStakeAmount

✓ should set min referrer stake amount correctly

setLastParticipationDate

✓ should set last participation date correctly (48ms)

updateStakingPower

✓ should update staking power correctly (496ms)

✓ should revert if id is invalid

setLastRegistrationDate

✓ should set last registration date correctly

stake

✓ should do stake correctly (858ms)

✓ should get info about stakes correctly (1138ms)

✓ should do stake correctly if not last tier (1332ms)

✓ should revert if amount is invalid in last tier (416ms)

✓ should revert if amount of stakeInfo is invalid where not last tier (460ms)

✓ should revert if amount of stake is zero

unstake

✓ should do unstake correctly (982ms)

✓ should do unstake correctly if stakingPower of caller more than zero (1050ms)

✓ should do unstake correctly if earlyExitFee more than zero (1193ms)

✓ should do unstake correctly if early exit (1325ms)

✓ should do unstake correctly with not receive booster (3079ms)

✓ should revert if id is zero

✓ should receive correct value of startBoosterInBP while unstake (1426ms)

stakeWithReferrer

✓ should do stake with referrer correctly (1104ms)

✓ should do stake with referrer correctly if referrer is zero's address (959ms)

✓ should update referral booster for stake correctly (2010ms)

✓ should receive correct referral booster while stake (2882ms)

✓ should revert if amount is zero

✓ should revert if amount is zero

```
    ✓ should revert if referring not valid
  stakeWithPermit
    ✓ should do stake with permit correctly (662ms)
  stakeWithPermitWithReferrer
    ✓ should do stake with permit with referrer correctly (1377ms)
  _authorizeUpgrade
    ✓ should revert if caller isn't admin (215ms)
  should calculate tierBoosterInBP correctly when:
    ✓ vesting period is zero (378ms)
    ✓ tier 1, vesting period 6 months (391ms)
    ✓ tier 2, vesting period 48 months (645ms)
    ✓ tier 3, vesting period 3 months (385ms)
  should calculate correctly when:
    3) should calculate referralBoosterInBP for parent correctly
       (stakeWithPermitWithReferrer)
    ✓ should calculate referralBoosterInBP for parent correctly (stakeWithReferrer) (2060ms)

StakingPool
  Check constructor
    ✓ should revert if registry zero address
    ✓ should revert if token zero address
    ✓ should revert emissionController zero address
    ✓ should set registry address
    ✓ should set token address
    ✓ should set emissionController address
  Check constructor
    mintForId
    ✓ should revert if not the manager
    ✓ should increase balance if _totalAmount > 0 (373ms)
    ✓ should increase totalSupply if _totalAmount > 0 (359ms)
    ✓ should increase idBalanceOf if _idAmount > 0 (209ms)
    ✓ should increase accountTotalSupply if _idAmount > 0 (223ms)
    ✓ should catch event Mint, and MintForId (220ms)
    burnForId
    ✓ should revert if not the owner
    ✓ should decrease balance if _totalAmount > 0 (488ms)
    ✓ should decrease totalSupply if _totalAmount > 0 (328ms)
    ✓ should decrease idBalanceOf if _idAmount > 0 (155ms)
    ✓ should decrease accountTotalSupply if _idAmount > 0 (254ms)
```

```
    ✓ should catch event BurnForId (241ms)
  withdraw
    ✓ should revert
  withdrawForId
    ✓ should revert
  withdrawForAccount
    ✓  should revert if invalid amount
    ✓ should withdraw all reward (589ms)
    ✓ should withdraw part of reward (539ms)
    ✓ should withdraw correctly with fee (640ms)

ThrottledPool
  Check constructor
    ✓ should revert if registry zero address
    ✓ should revert if token zero address
    ✓ should revert emissionController zero address
    ✓ should set registry address
    ✓ should set token address
    ✓ should set emissionController address
  Check functions:
    setEmissionController
    ✓ should revert if not the owner
    ✓ should revert emissionController zero address
    ✓ should catch event SetEmissionController
    setTokensPerSeconds
    ✓ should revert if not the owner
    ✓ should catch event SetTokensPerSeconds
    withdrawableRewardsForId
    ✓ should return correct withdrawable rewards if total reward > distributed reward (781ms)
    ✓ should return correct withdrawable rewards if total reward < distributed reward (482ms)
    ✓ should return correct withdrawable rewards (757ms)

124 passing (2m)
3 failing

1) IdPool
     Check functions:
      withdrawForId
        should update { tokenRewardsOf[account] & idTokenRewardsOf[account][_id] }:
```

```
    Error: VM Exception while processing transaction: reverted with panic code 0x11 (Arithmetic
operation underflowed or overflowed outside of an unchecked block)
      at IdPool._withdrawableRewardsOf (contracts/IdPool.sol:110)
      at IdPool.withdrawableRewardsForId (contracts/IdPool.sol:57)
      at processTicksAndRejections (internal/process/task_queues.js:97:5)
      at runNextTicks (internal/process/task_queues.js:66:3)
      at listOnTimeout (internal/timers.js:523:9)
      at processTimers (internal/timers.js:497:7)
      at HardhatNode.runCall (node_modules/hardhat/src/internal/hardhat-network/provider/
node.ts:534:20)
      at EthModule._callAction (node_modules/hardhat/src/internal/hardhat-network/provider/modules/
eth.ts:353:9)
      at HardhatNetworkProvider.request (node_modules/hardhat/src/internal/hardhat-network/
provider/provider.ts:117:18)
      at EthersProviderWrapper.send (node_modules/hardhat-deploy-ethers/src/internal/ethers-provider-
wrapper.ts:13:20)

  2) IdPool
       Check functions:
         withdrawableRewardsForId
           should return correct withdrawable rewards:
     AssertionError: Expected "550000000000000000000" to be equal 10750000000000000000000
      at /home/romario/Work/Audit/UpLift-last-test/staking-uplift/test/IdPool.test.ts:371:88
      at step (test/IdPool.test.ts:52:23)
      at Object.next (test/IdPool.test.ts:33:53)
      at fulfilled (test/IdPool.test.ts:24:58)
      at processTicksAndRejections (internal/process/task_queues.js:97:5)

  3) Staking
       Check functions:
         should calculate correctly when:
           should calculate referralBoosterInBP for parent correctly (stakeWithPermitWithReferrer):

      AssertionError: expected 0 to equal 1
      + expected - actual

      -0
      +1
```

```
at /home/romario/Work/Audit/UpLift-last-test/staking-uplift/test/Staking.test.ts:1291:83
at step (test/Staking.test.ts:52:23)
at Object.next (test/Staking.test.ts:33:53)
at fulfilled (test/Staking.test.ts:24:58)
at runMicrotasks (<anonymous>)
at processTicksAndRejections (internal/process/task_queues.js:97:5)
at runNextTicks (internal/process/task_queues.js:66:3)
at listOnTimeout (internal/timers.js:523:9)
at processTimers (internal/timers.js:497:7)
```

We are grateful to have been given the opportunity to work with the Uplift team.

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.

Zokyo's Security Team recommends that the Uplift team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.