

Assure DeFi[®]

THE VERIFICATION **GOLD STANDARD**



Security Assessment

ROAI

Date: 14/06/2025

Audit Status: PASS

Audit Edition: Advanced+



ASSURE DEFI[®]
THE VERIFICATION **GOLD STANDARD**

Risk Analysis

Vulnerability summary

Classification	Description
 High	High-level vulnerabilities can result in the loss of assets or manipulation of data.
 Medium	Medium-level vulnerabilities can be challenging to exploit, but they still have a considerable impact on smart contract execution, such as allowing public access to critical functions.
 Low	Low-level vulnerabilities are primarily associated with outdated or unused code snippets that generally do not significantly impact execution, sometimes they can be ignored.
 Informational	Informational vulnerabilities, code style violations, and informational statements do not affect smart contract execution and can typically be disregarded.

Executive Summary

According to the Assure assessment, the Customer's smart contract is **Secured**.



Scope

Target Code And Revision

For this audit, we performed research, investigation, and review of the ROAI contracts followed by issue reporting, along with mitigation and remediation instructions outlined in this report.

Target Code And Revision

Project	Assure
Language	Solidity
Codebase	roai-staking-contracts.zip [SHA256]: f364fddf5ed109077de9daf56722de32cf5af16a6cfbdd4732e33c6e4f3f41ca Fixed version - Revised contracts.zip [SHA256]: 5019c0c933f4b556e8167e3cda8c46558bd31b93646e94e436af64606a11cbc7
Audit Methodology	Static, Manual

Attacks made to the contract

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices.

Category	Item
Code review & Functional Review	<ul style="list-style-type: none">• Compiler warnings.• Race conditions and Reentrancy. Cross-function race conditions.• Possible delays in data delivery.• Oracle calls.• Front running.• Timestamp dependence.• Integer Overflow and Underflow.• DoS with Revert.• DoS with block gas limit.• Methods execution permissions.• Economy model.• Private user data leaks.• Malicious Event log.• Scoping and Declarations.• Uninitialized storage pointers.• Arithmetic accuracy.• Design Logic.• Cross-function race conditions.• Safe Zeppelin module.• Fallback function security.• Overpowered functions / Owner privileges

AUDIT OVERVIEW



1. Batch-DoS via Malicious Pool [Fixed ✓]

Function: ROAIStakingFactory.distributeRewardsToMultiplePools

Issue: The factory loops through `_poolAddresses` and performs an external call to each:

```
(bool success, ) = poolAddress.call{value: amount, gas: gasToUse}("");  
if (!success) { revert(...) }
```

If any one pool rejects the ETH transfer (e.g. via a revert in its fallback/receive), the entire batch distribution reverts and no pool receives rewards.

Recommendation: Use try/catch or low-level calls that don't revert the loop Or wrap the call in an unchecked block and emit an event on failure without reverting the entire function.

Fix: The Batch-DoS vulnerability has been properly addressed in the current implementation. The use of try/catch ensures that a single malicious pool cannot prevent rewards from being distributed to other valid pools. The function will now complete successfully for all valid pools even if some pools fail to accept the transfer.

2. Lost Rewards on Unstake [Fixed ✓]

Function: ROAIStakingPool / unstake

Issue: `unstake()` zeroes out `position.amount` and marks `position.active = false` before any reward payout, then only transfers the staked tokens. Although it does call `updateRewards`, it never actually sends accrued `accumulatedRewards` to the user. Afterwards, `claimRewards()` will revert because `position.active == false`.

Recommendation: In `unstake`, if `position.accumulatedRewards > 0`, send those ETH immediately to the user (e.g. via the same `call{value:}`), and decrement `totalRewards`.

Or else allow `claimRewards` to work on inactive positions by removing the `require(position.active,...)` check in `claimRewards`, also emit a combined `UnstakedAndRewardPaid` event for clarity.

Fix: The fix implements the recommended solution by:

Calculating and preserving rewards before marking the position inactive

Using a pull-based reward system

Properly emitting events to track reward payments



MEDIUM

1. Missing AccessControl Validation [Fixed ✓]

Function: ROAIStakingFactory.createStakingPool

Issue: If the factory owner calls createStakingPool before having set the accessControl address, the newly deployed pool will receive address(0) for its AccessControl parameter. Depending on pool implementation, this may disable role checks, allow unauthorized actions, or break pool logic entirely.

Recommendation:

```
require(accessControl != address(0), "AccessControl not set");
```

Insert this check at the top of createStakingPool. Additionally, document that setAccessControl must be called before any pool creation.

Fix: The issue has been fully addressed. The implementation matches the recommendation exactly, including:

The explicit require check

Proper placement of the check

Clear error messaging

Documentation via code comments

2. Checks-Effects-Interactions Pattern [Partially Solved ✓]

Function: ROAIStakingFactory.distributeRewardsToMultiplePools

Issue: External calls to poolAddress occur before any potential state changes or logging, violating the Checks-Effects-Interactions pattern. Also ETH transfers to staking pools are done without any reentrancy guard. A malicious pool whose fallback reenters.

Recommendation: Reorder logic so that all internal state validations and updates occur before external calls. If logs are critical, emit an event after state updates but before the call, then revert on failure only of that call if appropriate. Consider adding a reentrancyGuard.

Fix: Input validations are properly done before any interactions but no reentrancy guard modifier is used.

3. Stale/Destroyed-pool Hijack [Fixed ✓]

Contract: ROAIStakingFactory

Issue: Even after a staking pool is destroyed (self-destructed) or replaced by an EOA, isStakingPool[pool] remains true. Subsequent distributions will send ETH to attacker-controlled addresses.

Recommendation: require(poolAddress.code.length > 0, "Pool no longer alive");

Fix: Fixed by the existing code.length check.

4. Push-Payment DoS/Theft Vector [Acknowledge]

Contract: ROAIStakingFactory

Issue: "Push" payments via .call{value} let recipients revert, grief gas, or pre-self-destruct, causing batch failure or outright ETH theft when a malicious recipient traps the funds.

Recommendation: Adopt a pull-based pattern:

Maintain mapping(address => uint256) pendingRewards;

In the distribution loop, do pendingRewards[pool] += amount; (no ETH transfers).

Expose function withdrawRewards() external nonReentrant {} so each pool pulls its owed ETH safely.

5. Non-Safe ERC-20 Interactions [Fixed ✓]

Contract: ROAStakingPool (Transferfrom,transfer,emergencyWithdrawTokens)

Issue: Using raw IERC20 without OpenZeppelin's SafeERC20 means missing bool return values or non-standard tokens could cause silent failures or reverts.

Recommendation: Integrate OpenZeppelin's SafeERC20 library and replace calls with SafeERC20.safeTransfer/safeTransferFrom.

Fix: The contract now securely handles ERC20 transfers using OpenZeppelin SafeERC20

6. Unbounded Growth of User Positions (DoS) [Fixed ✓]

Contract: ROAStakingPool (Unstake/Stake)

Issue: Every time a user stakes after fully unstaking, a new entry is appended to stakingPositions[msg.sender]. Over time, a single address can accumulate an unbounded number of historical positions. The unstake function then loops over all of them to check for any remaining active positions, causing gas cost to grow linearly with the number of past stakes.

Recommendation: Cap the number of positions per address (for example max 5).

Maintain a counter of active positions and decrement it on unstake—so you don't need to loop through the entire array and optionally, prune or overwrite old completed positions instead of appending indefinitely

Fix: Unbounded position growth is fixed via:

Hard cap (MAX_POSITIONS_PER_USER = 1).

Automatic pruning of inactive positions



1.CREATE2 Salt Predictability [Fixed ✓]

Function: ROAStakingFactory.createStakingPool

Issue: The salt is computed as

```
keccak256(abi.encodePacked(
    msg.sender,
    _poolCreationNonce,
    _name,
    block.timestamp,
    blockhash(block.number - 1)
))
```

While incorporating blockhash and a nonce, an attacker observing a pending transaction can partially

predict the salt (since `blockhash(block.number - 1)` and `block.timestamp` have bounded entropy), enabling them to precompute the pools address off-chain

Recommendation: Augment salt with a truly secret or user-provided value, or require the factory owner to provide a random “creatorSalt” parameter when deploying each pool.

Fix: The salt now includes 11 variables



INFORMATIONAL

1. Stuck ETH from Self-Destruct Attacks [Acknowledge]

Contract: ROAIStakingPool (receive() fallback and reward logic)

Issue: If ETH is sent via selfdestruct (bypassing receive), totalRewards is not incremented even though contract balance increases.

Recommendation: Add a fallback() that also credits totalRewards or periodically reconcile totalRewards = address(this).balance in maintenance.

2. Emergency Admin Can Drain Staked ROAI [Acknowledge]

Contract: ROAIStakingPool (emergencyWithdraw, emergencyWithdrawAllETH)

Issue: The EMERGENCY_ADMIN_ROLE (or the accessControl.owner()) can call emergencyWithdrawTokens(address _roaiToken, uint256 _amount) to pull any ERC-20 including the very ROAI tokens staked by users directly to the owner. There is no distinction between reward tokens and staking tokens.

Recommendation: Restrict emergencyWithdrawTokens so it cannot withdraw the pool designated roaiToken (e.g. require(_tokenAddress != address(roaiToken))).

If true emergency recovery is needed, introduce a time-lock or multi-sig delay for critical withdrawals of the staking token.

Testing coverage

During the testing phase, custom use cases were written to cover all the logic of contracts. **Check “Annexes” to see the testing code.*

```
tests/test_staking_factory.py::test_set_access_control RUNNING
Transaction sent: 0x1f4881de550631689b4be6739dc8f6561f8c11e7e14c4ccbf6a91bd9ac2574
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 12
ERC20Mock.constructor confirmed Block: 14 Gas used: 619603 (5.16%)
ERC20Mock deployed at: 0x2c15A156108fa5248E4Cb0bd693320e908E03Cc

Transaction sent: 0xaf20b7d86e70f0cf46f39f630bb6f8aff54596db5b9f74eb127339ba664db541
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 13
AccessControl.constructor confirmed Block: 15 Gas used: 2529423 (21.08%)
AccessControl deployed at: 0xe692Cf21B12e082717C4bF647F9768Fa58861c8b

Transaction sent: 0xfefaa5cbffa58ce78b0567a9d739601dbe293d984167b66042d8bccd3ef473c1
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 14
R0A1StakingFactory.constructor confirmed Block: 16 Gas used: 4766889 (39.72%)
R0A1StakingFactory deployed at: 0xe65A7a341978d59d40d30FC23F5014FACB4f575A

Transaction sent: 0x8cb083aa7df614882f68afc491d2870c2db19c515418fafd5d03523f0ded2e59
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 1
R0A1StakingFactory.setAccessControl confirmed (Ownable: caller is not the owner) Block: 17 Gas used: 22744 (0.19%)

Transaction sent: 0x76a7f3146702e65eeff18f1ced15ecab1662aff593d1822f86d698b81bfd28f8
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 15
R0A1StakingFactory.setAccessControl confirmed (Invalid AccessControl address) Block: 18 Gas used: 22534 (0.19%)

Transaction sent: 0x081ffdd78738b20659d1853f9ba2014f253fda4522818227ff3d3f2ed2b4274a
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 16
R0A1StakingFactory.setAccessControl confirmed Block: 19 Gas used: 44601 (0.37%)

tests/test_staking_factory.py::test_set_access_control PASSED
tests/test_staking_factory.py::test_distribute_rewards RUNNING
Transaction sent: 0xe26d48ddde9b3d857b2445e0de7bf57ab1960121af1a4670c30c6b1cf8a74eeb
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 17
ERC20Mock.constructor confirmed Block: 20 Gas used: 619603 (5.16%)
ERC20Mock deployed at: 0xf80588c72B438faD4Cf7cD879c8F730Faa2130a0

Transaction sent: 0xc4076f30590b7ad6ae4d52aa7ded2eb8967f6f281651a905f1522abdelaa46df2
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 18
R0A1StakingFactory.constructor confirmed Block: 21 Gas used: 4766889 (39.72%)
R0A1StakingFactory deployed at: 0xed00238F9A0F7b4d93842033cdF56cCB32C781c2

Transaction sent: 0xb77f175c12dad22e9fbac8608c4a7b1cf6995480053de154087bbcf328266c1d
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 19
R0A1StakingFactory.setRewardPool confirmed Block: 22 Gas used: 44622 (0.37%)

Transaction sent: 0xf89e2ad1936f617a6fcace764a640bf2cf41066a6daa05c589dclb8088b49bcbf
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 20
R0A1StakingFactory.distributeRewardsToMultiplePools confirmed (Only reward pool can distribute) Block: 23 Gas used: 23511 (0.20%)

Transaction sent: 0x9a38320da321734427b2b6bab6befcc42a3c718353e872607994589b38dda6d
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 0
R0A1StakingFactory.distributeRewardsToMultiplePools confirmed (Percentages must sum to 100) Block: 24 Gas used: 23592 (0.20%)

Transaction sent: 0x72c6ebc663ece84063d2aa8aa5a9e643889d539c1248f15bce22dd6c596e326d
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 1
R0A1StakingFactory.distributeRewardsToMultiplePools confirmed (Arrays must be same length) Block: 25 Gas used: 23686 (0.20%)

Transaction sent: 0xe927b23e232668c7b1e2991066f4fe5d2ba4d0fe4af3ccf4e4067f53e2761071
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 2
R0A1StakingFactory.distributeRewardsToMultiplePools confirmed (Not a valid staking pool) Block: 26 Gas used: 25470 (0.21%)

Transaction sent: 0x8del292204c1e02f462c5d3a88995a02f611224327753811e4f7153ad4d65f3a
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 21
R0A1StakingFactory.createStakingPool confirmed Block: 27 Gas used: 3731708 (31.10%)

Transaction sent: 0xc291fed35577a24de225918cbd80eb4f0c276ebff9341ec36006435374f99e00
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 3
R0A1StakingFactory.distributeRewardsToMultiplePools confirmed (ETH transfer failed to pool) Block: 28 Gas used: 33427 (0.28%)

Transaction sent: 0x91dede4c8d1267e4618d3165a7a8c1133b37810e3994ee4ba8cc0f139bfe49fa
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 4
R0A1StakingFactory.distributeRewardsToMultiplePools confirmed Block: 30 Gas used: 58450 (0.49%)

tests/test_staking_factory.py::test_distribute_rewards PASSED
```

```
tests/test_staking_pool.py::test_constructor RUNNING
Transaction sent: 0xbfa9f1f864d99e64a5846a3a59398bed9e682643c84e7a5cb7345f55ce263ca2
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 0
ERC20Mock.constructor confirmed Block: 1 Gas used: 619683 (5.16%)
ERC20Mock deployed at: 0x3194c80d3dbcd3e11a87892e7bA5c3394848Cc87

Transaction sent: 0x5cbeb3411cfa7e4853775efa09211fda298d2ba7a80f843e096d1f3231981c0c
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 1
AccessControl.constructor confirmed Block: 2 Gas used: 2529423 (21.88%)
AccessControl deployed at: 0x682C71e40AC47a842Ee7f46E8aee17F94A3bA086

Transaction sent: 0x87d3aa55a3387b82e1fff7a843e15f1935452aa1e8018a9dc43ba7da42380064
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 2
ROAStakingPool.constructor confirmed (Invalid token address) Block: 3 Gas used: 372606 (3.11%)

Transaction sent: 0xb7b7754d45e4f36ce8f6551alc276a118eb64527252e7388a04f59f14c39f5c1
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 3
ROAStakingPool.constructor confirmed (Invalid max stakers) Block: 4 Gas used: 372961 (3.11%)

Transaction sent: 0xaa8d8264f1e491867daca2c567b81642a7d374e43862d38efe2fd750851121
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 4
ROAStakingPool.constructor confirmed (Invalid stake amount) Block: 5 Gas used: 372960 (3.11%)

Transaction sent: 0xc0f572bb46cc0f2a1c98687b2ac4d5d0bc3f80593e2c5e70c3f20f05f01f28
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 5
ROAStakingPool.constructor confirmed Block: 6 Gas used: 3887387 (32.39%)
ROAStakingPool deployed at: 0x6b480e1086912A6Cb24ce3d843b3466ebc72AFd3

tests/test_staking_pool.py::test_constructor PASSED
```

```
tests/test_staking_pool.py::test_stake RUNNING
Transaction sent: 0x3cc3d2e985d3e7baa428136656f485f5d81a74f23c02c9c1d2bf0964f086eeb
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 6
ERC20Mock.constructor confirmed Block: 7 Gas used: 619683 (5.16%)
ERC20Mock deployed at: 0x9E4c14483d7d9A8A752044E86a93CAE090782ac9

Transaction sent: 0x60eb85a010ddcba06e183615b7569d48acc4447a6ef9f332c4419cd46d14d8ea
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 7
AccessControl.constructor confirmed Block: 8 Gas used: 2529423 (21.88%)
AccessControl deployed at: 0xc053c9429d32594f484d01f0e9E65ED10Cd809

Transaction sent: 0xdff4e7964fe7ca3eb5887ac13b73e0067d72d065d1718f18a32ce91bae01e48a1
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 8
ROAStakingPool.constructor confirmed Block: 9 Gas used: 3887375 (32.39%)
ROAStakingPool deployed at: 0x428b109989ef5baba6092829594ef45E19A04AAA

Transaction sent: 0xf82cd6ca67b81488a83ff8b4aec69e89719d14a54ac268af563fcc7bef856631
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 9
ROAStakingPool.stake confirmed (Invalid lock period index) Block: 10 Gas used: 29166 (0.24%)

Transaction sent: 0xe23e18aed61c47d68c874ca14ff3bcd18f2109720558779598365149f2f81
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 1
ROAStakingPool.disableLockPeriod confirmed (Access denied: Pool Manager role required) Block: 11 Gas used: 30526 (0.25%)

Transaction sent: 0xc3d93b4053b325d337b56f3d3a9f14f72c5821d8d179f5f1cd15c445bfd56541
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 9
ROAStakingPool.disableLockPeriod confirmed Block: 12 Gas used: 20669 (0.17%)

Transaction sent: 0x00e4ebf377552ecc6d5b0f58429b618dc2db23f356c448d8b82e8866257b34
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 2
ROAStakingPool.stake confirmed (Lock period not active) Block: 13 Gas used: 30868 (0.26%)

Transaction sent: 0x7b98915c276d2f16f60723e87f3a5b22a7057a9bcaaa44a3dbf909a10b7cd10f
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 10
ERC20Mock.mint confirmed Block: 14 Gas used: 65613 (0.55%)

Transaction sent: 0xf7681ade2178afd3cdac54310c6ff7005d17a0cc2cf9eb956820c78cbb5e2e2
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 11
ERC20Mock.mint confirmed Block: 15 Gas used: 50601 (0.42%)

Transaction sent: 0xd9cd28926e3f45a736c20287a1ee18580a2f650295call135f1d0e47d7774360e
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 12
ERC20Mock.mint confirmed Block: 16 Gas used: 50613 (0.42%)

Transaction sent: 0x4919304912638e32edeecc8da31825684749da9a55a44be406af80f5e472f82c9
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 3
ERC20Mock.approve confirmed Block: 17 Gas used: 44136 (0.37%)

Transaction sent: 0xd431924ac1aalcabcb08bd283686d4aaac82b23f09f205f5dafc7055f4acd6d4
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 0
ERC20Mock.approve confirmed Block: 18 Gas used: 44136 (0.37%)

Transaction sent: 0x9fc7d768173042692577bb73871ef61f9be7184815c26c792722aeb53f5eb6b
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 0
ERC20Mock.approve confirmed Block: 19 Gas used: 44136 (0.37%)

Transaction sent: 0x74804b3c10f5f3d840e64429cd891c4ded389f0e07afdb17b8b8c83de60cbb58
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 4
ROAStakingPool.stake confirmed Block: 20 Gas used: 30566 (2.55%)

Transaction sent: 0x84269b34bc9e42d34fd445bebca2f90763d76bd5ad17cb394bd2b6a84db49b72
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 5
ROAStakingPool.stake confirmed (User has already staked in this pool) Block: 21 Gas used: 33415 (0.28%)

Transaction sent: 0x57a3af33eda5d7716339ca725a38639204abbce2da8c2f1ad6bc42f062102d46b
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 1
ROAStakingPool.stake confirmed Block: 22 Gas used: 20066 (2.17%)

Transaction sent: 0x84cf529e2ce5522c3e7fbee1c2c816ef7b94412a98e63b3eef0b92bcadec510
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 1
ROAStakingPool.stake confirmed (Pool is full) Block: 23 Gas used: 32503 (0.27%)

tests/test_staking_pool.py::test_stake PASSED
```

```

tests/test_staking_pool.py::test_ustake RUNNING
Transaction sent: 0x77b19c99b4169c289b3847621f5ed2f7845581401f1373fa8893fab96b13cc86
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 13
ERC20Mock.constructor confirmed Block: 24 Gas used: 619603 (5.16%)
ERC20Mock deployed at: 0xe692cf21812e882717C4bf647F9768Fa58861c8b

Transaction sent: 0x18b478da2feceaa6319933437d793d7d6072122058b8c4625f72452c14b2b390
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 14
AccessControl.constructor confirmed Block: 25 Gas used: 2529423 (21.00%)
AccessControl deployed at: 0xe65a7a341978d59d40d30fc23f5014fACB4f575A

Transaction sent: 0xf7246b92dbca932a6245868ea71aa168fef2dd970520ff9b9c5573541b1b5e70
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 15
ROAStakingPool.constructor confirmed Block: 26 Gas used: 3887375 (32.39%)
ROAStakingPool deployed at: 0x303758532345801c88c2A012541b09E9Aa53A93d

Transaction sent: 0x190dc85133ba43b476a19a9305432a5883b0a06e5f83452ca61ca389e06a8807
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 16
ERC20Mock.mint confirmed Block: 27 Gas used: 65613 (0.55%)

Transaction sent: 0xd5880fa3ea21e98f5832abcd69d357b9d8ca840c3d9cf9593d0edb982ddf0e9f
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 17
ERC20Mock.mint confirmed Block: 28 Gas used: 50601 (0.42%)

Transaction sent: 0x36f3944c6deee873c48d5ec008a2457048123928cf4a23b110a5fe8228f00de3
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 6
ERC20Mock.approve confirmed Block: 29 Gas used: 44136 (0.37%)

Transaction sent: 0xcb50000f082f503d72d2bfaa86e0f37c4bc564766bb29122e6fcab0f181da9c1
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 2
ERC20Mock.approve confirmed Block: 30 Gas used: 44136 (0.37%)

Transaction sent: 0xb13f2a7e8e4a483e4294d36972617720854dc5edcbde41de2cd5d98acd6a1876
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 7
ROAStakingPool.stake confirmed Block: 31 Gas used: 305646 (2.55%)

Transaction sent: 0x6834e793094f06d96c6ea2f2b0182f387012ca8ee8cfb89cf6fd97399a31bfc4
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 3
ROAStakingPool.stake confirmed Block: 32 Gas used: 200646 (2.17%)

Transaction sent: 0x5034398a079ea81928c7d6340655636796653dd62af6a07b245f45cb29c0b994
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 8
ROAStakingPool.unstake confirmed (Invalid position index) Block: 33 Gas used: 29175 (0.24%)

Transaction sent: 0x4c080b6b222e3c61c3f6216257650ea540fab38c311b4670cd566047a7d72f4c
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 9
ROAStakingPool.unstake confirmed (Lock period not ended) Block: 34 Gas used: 31784 (0.26%)

Transaction sent: 0x5e271eae03a4b121adb8c3acb98275a3fe0adce2c2167bc6bd62c4ea6e7b5379
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 10
ROAStakingPool.unstake confirmed Block: 37 Gas used: 64375 (0.54%)

tests/test_staking_pool.py::test_ustake PASSED

```

```

tests/test_staking_pool.py::test_claim_rewards RUNNING
Transaction sent: 0xf122644a0c79d3916500a69591b88b23a0854544221fcd834de08da27638f
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 19
ERC20Mock.constructor confirmed Block: 38 Gas used: 619603 (5.16%)
ERC20Mock deployed at: 0x0ae02e4fe488952cf88c951771540188647a0146

Transaction sent: 0x35bbdb3fedb16dbfb206f5b0152dcd02c35567b6da6d8b7bb8d34b3c8976998a
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 20
AccessControl.constructor confirmed Block: 39 Gas used: 2529423 (21.08%)
AccessControl deployed at: 0xdCF93F11ef216cEC9C07fd31d0801c9b2b39Afb4

Transaction sent: 0x2b5bf0327a304a6605dd7328777374d44cc9feb2cle93b04c54c944829015327
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 21
ROAStakingPool.constructor confirmed Block: 40 Gas used: 3887375 (32.39%)
ROAStakingPool deployed at: 0xBcb61491F1859f53438918F1A5aFCA542A190397

Transaction sent: 0xdd617c80804f8ad3654688a73401c6fc671cae999e96ec8235459855f908630a
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 22
ERC20Mock.mint confirmed Block: 41 Gas used: 65613 (0.55%)

Transaction sent: 0x3a058ffc7b02c4203f24ea2964c33751a895b5837f869d90e76d84bbcb275e7
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 23
ERC20Mock.mint confirmed Block: 42 Gas used: 50601 (0.42%)

Transaction sent: 0x5b1666a1b7b8a0ad6aef9e3cbbafcb0a8ed70daa5dc69f5d12165f75ff496038
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 24
ERC20Mock.mint confirmed Block: 43 Gas used: 50613 (0.42%)

Transaction sent: 0x4380fc9c3fd2504850e89ccfc7e76c92c43e82162ddc629f1fc6cabcb688c2
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 11
ERC20Mock.approve confirmed Block: 44 Gas used: 44136 (0.37%)

Transaction sent: 0x33fc59fa98dbff91a9d5de20c7c015706ae261fb0756f12ced7a8590889f2cfe
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 4
ERC20Mock.approve confirmed Block: 45 Gas used: 44136 (0.37%)

Transaction sent: 0xee3b48a6c3544a9116d90242cfbc989487f2fba35a4ef9329b83ec3639cc7e5
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 2
ERC20Mock.approve confirmed Block: 46 Gas used: 44136 (0.37%)

Transaction sent: 0xd5721d216d38aa2a8fa14a5c584766848834b179bdca834c52c53bf496dc3
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 12
ROAStakingPool.stake confirmed Block: 47 Gas used: 38566 (2.55%)

Transaction sent: 0x1d13bf70d9cb02a57beb60169223353e64e0d3c14e7a6e8c36eff59137338a36
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 5
ROAStakingPool.stake confirmed Block: 48 Gas used: 260646 (2.17%)

Transaction sent: 0xea2da051b4e0d1ba9b325a63f7d6d21af4dbcc63728db0e0c5ab8490cf408dc2
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 3
ROAStakingPool.stake confirmed Block: 49 Gas used: 260646 (2.17%)

Transaction sent: 0x920dd81c334f41e43669b02752cf9df435b907c95c4a6d08228af4b0e8923509
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 13
ROAStakingPool.claimRewards confirmed (Invalid position index) Block: 51 Gas used: 29242 (0.24%)

Transaction sent: 0x284a98058b329dd917a53b76317d1347bf7b8e7d8178cde572bcd0f33efb313a
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 14
ROAStakingPool.claimRewards confirmed (No rewards to claim) Block: 52 Gas used: 36323 (0.30%)

Transaction sent: 0x66b4832972e04fd8e67b26ee590c75b983f6fc77d6e9d3ea41bb2f7fa904ebf6
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 15
ROAStakingPool.claimRewards confirmed Block: 54 Gas used: 64426 (0.54%)

Transaction sent: 0x669fa33a67fc232ecfc2046ba5b5e860196e4d7d1e65b33ff6177cddbf4cea94
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 6
ROAStakingPool.claimRewards confirmed Block: 56 Gas used: 64426 (0.54%)

tests/test_staking_pool.py::test_claim_rewards PASSED

```

```

tests/test_reward_distributor.py::test_distributeRewards RUNNING
Transaction sent: 0x7d9377cdcaff252e79e57d7d9e64861bbbcbace5c5d3a00c96ddfa99e2f2d2eb
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 0
SimpleRewardDistributor.constructor confirmed Block: 1 Gas used: 1372072 (11.43%)
SimpleRewardDistributor deployed at: 0x3194c80c3dbcd3E11a07892e7bA5c3394048Cc87

Transaction sent: 0x2376676e819534c8e817848f2a4d24e6638fc659e912ae01d187c1ab5d858b01
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 0
SimpleRewardDistributor.distributeRewards confirmed (Ownable: caller is not the owner) Block: 2 Gas used: 23334 (0.19%)

Transaction sent: 0xe941ca2e73598d60620d69b5e37ac35851795d154baf03afdc2b692b565f3699
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 1
SimpleRewardDistributor.distributeRewards confirmed (No pools specified) Block: 3 Gas used: 29216 (0.24%)

Transaction sent: 0xa68df63d13dd9496cb6a93084e1048204688cea87d3f9940b54a11f3814738a
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 2
SimpleRewardDistributor.distributeRewards confirmed (Arrays length mismatch) Block: 4 Gas used: 29899 (0.25%)

Transaction sent: 0x2bddd0fbd05f6d9a2cf3c935953b5938ed65d0300a1c35a869df66992a08bf
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 3
SimpleRewardDistributor.distributeRewards confirmed (Percentages must sum to 100) Block: 5 Gas used: 29988 (0.25%)

Transaction sent: 0x1effe35759c8249ffb4d8a618771ff39b3d4aeeef99e9ed71e8159d247d5c1a
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 4
SimpleRewardDistributor.distributeRewards confirmed (No funds available for distribution) Block: 6 Gas used: 31129 (0.26%)

Transaction sent: 0x028cfc65b9a77dac5d2c9a619256d6197e2b38cfa5436be100f7788d7158ea
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 6
SimpleRewardDistributor.distributeRewards confirmed (Invalid pool address) Block: 8 Gas used: 31381 (0.26%)

Transaction sent: 0xd7b0980f1e1fadf6249a27bc0fecb30ab5c5a67c2c4a2f7bca5804278a382fd5
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 7
SimpleRewardDistributor.distributeRewards confirmed Block: 9 Gas used: 59139 (0.49%)

tests/test_reward_distributor.py::test_distributeRewards PASSED
tests/test_reward_distributor.py::test_set_reserve_perc RUNNING
Transaction sent: 0xa08626fdb52c6f17c7bca17edafa88679600ac3db2445c8b6e78fd0d348bfa71
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 8
SimpleRewardDistributor.constructor confirmed Block: 10 Gas used: 1372072 (11.43%)
SimpleRewardDistributor deployed at: 0x420b109989eF5baba6092029594eF45E19A044AA

Transaction sent: 0x462669350e7933d28c30fc7f54c83407b98b451745a3dff9b84df39c85cf38e
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 1
SimpleRewardDistributor.setReservePercentage confirmed (Ownable: caller is not the owner) Block: 11 Gas used: 22436 (0.19%)

Transaction sent: 0x70e9475fe5d4073b60228ae8520a8fbd1efa28b3cfc45c611cf28cf5b6b09bb
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 9
SimpleRewardDistributor.setReservePercentage confirmed (Reserve cannot exceed 50%) Block: 12 Gas used: 22468 (0.19%)

Transaction sent: 0x134da743fc442ddca7b3d8b5522e2594004e3bee89bba89d0d6df7b7523e8f2
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 10
SimpleRewardDistributor.setReservePercentage confirmed Block: 13 Gas used: 29514 (0.25%)

tests/test_reward_distributor.py::test_set_reserve_perc PASSED
tests/test_reward_distributor.py::test_set_treasury RUNNING
Transaction sent: 0x1826560fcd757153313b9277a6a26a2bc6ecea557bdae20d1f9bc7f5e700523
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 11
SimpleRewardDistributor.constructor confirmed Block: 14 Gas used: 1372072 (11.43%)
SimpleRewardDistributor deployed at: 0x7a3d735ee6873f170bdcab1d518604928dc10d92

Transaction sent: 0x84496711e40ca91836af0fdca3cae2891605cb06c014026571bbd6204195d1d4
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 2
SimpleRewardDistributor.setTreasuryMallet confirmed (Ownable: caller is not the owner) Block: 15 Gas used: 22747 (0.19%)

Transaction sent: 0x54097c817ee6f4d5bda004b6ae05987bd3e353c0191e6916e1ff195c9872a341
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 12
SimpleRewardDistributor.setTreasuryMallet confirmed (Invalid treasury address) Block: 16 Gas used: 22560 (0.19%)

Transaction sent: 0x721db68f12e62a39c7e9abb3332206dcb3ea20c9d31d07c1155c074ba88f1bd
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 13
SimpleRewardDistributor.setTreasuryMallet confirmed Block: 17 Gas used: 30087 (0.25%)

tests/test_reward_distributor.py::test_set_treasury PASSED

```



```
tests/test_staking_factory.py::test_create_staking_pool RUNNING
Transaction sent: 0xbfa9f1f664d99e64a584663a59398bed9e602643c84e7a5cb7345f55ce263ca2
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 0
ERC20Mock.constructor confirmed Block: 1 Gas used: 619683 (5.16%)
ERC20Mock deployed at: 0x3194c80C3dbcd3E11a87892e7bA5c3394048Cc87

Transaction sent: 0x5cbeb3411cfa7e4853775efa09211fda298d2ba7a80f843e096d1f3231981c0c
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 1
AccessControl.constructor confirmed Block: 2 Gas used: 2529423 (21.08%)
AccessControl deployed at: 0x602C71e40AC47a042Ee7f46E0ae17F94A3bA086

Transaction sent: 0xb14683841e05a01d9ea0c8681ac320f20bdc953566d7488e1721d97e954ed9ba
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 2
ROAStakingFactory.constructor confirmed Block: 3 Gas used: 4766889 (39.72%)
ROAStakingFactory deployed at: 0xE7e06747FaC5360f88a2EFC03E00d25789F69291

Transaction sent: 0x26157f8824eb892c28b1f2c88a76b683c72b9b9f6438b85cdd2902b81a0bf65c
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 3
ROAStakingFactory.createStakingPool confirmed (Invalid max stakers) Block: 4 Gas used: 23502 (0.20%)

Transaction sent: 0x63dafa6a6c6da7f46cb481ecab882547368817723e816a302801649bce3e9e7
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 4
ROAStakingFactory.createStakingPool confirmed (Invalid stake amount) Block: 5 Gas used: 23513 (0.20%)

Transaction sent: 0x263cfaab7cd0cab8e28b53f105b96ae416019dddc541c0fd71d617a34e8b263
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 5
ROAStakingFactory.createStakingPool confirmed (Invalid pool name) Block: 6 Gas used: 23505 (0.20%)

Transaction sent: 0x1119d30b21cf25206a0311c25bc4999901f1105bea326c1f14cc7aa5cdddf96e
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 6
ROAStakingFactory.createStakingPool confirmed Block: 7 Gas used: 3731708 (31.10%)

tests/test_staking_factory.py::test_create_staking_pool PASSED
tests/test_staking_factory.py::test_set_reward_pool RUNNING
Transaction sent: 0x563a398c295acfb05b2dff66c154cb006dc393a3150446a50b88d2f3b5ba5711
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 7
ERC20Mock.constructor confirmed Block: 8 Gas used: 619683 (5.16%)
ERC20Mock deployed at: 0xcCB53c9429d32594F484d01f0e9E65ED1DCda809

Transaction sent: 0x21bd144182ecbe4a7b3dfbec1b462935f39eb1be8dbadaf060f0a45a9b6848df
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 8
AccessControl.constructor confirmed Block: 9 Gas used: 2529423 (21.08%)
AccessControl deployed at: 0x420b109909eF5baba6092029594eF45E19A04AAA

Transaction sent: 0x4e6a34868f0661892321bdd49d8fdb60c483fa0899d84eb5bb4c0161c135099b
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 9
ROAStakingFactory.constructor confirmed Block: 10 Gas used: 4766889 (39.72%)
ROAStakingFactory deployed at: 0xa3853d0Cd2E3fC28e8E130288F2a808d5EE37472

Transaction sent: 0xc198473b52e1c4bd98dcddddd1357b051f61593c4f9c3b382601dac3f4c08ead5
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 0
ROAStakingFactory.setRewardPool confirmed (Ownable: caller is not the owner) Block: 11 Gas used: 22754 (0.19%)

Transaction sent: 0x55a0e68a0eb23c96ec163b25cadb36d51c06dc8ec8525a156678654e1a60fab0
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 10
ROAStakingFactory.setRewardPool confirmed (Invalid reward pool address) Block: 12 Gas used: 22567 (0.19%)

Transaction sent: 0xaeae0ef9b05a078b7351a72f2673c045f18b09cffa7be9222e1f3afd48453613
Gas price: 0.0 gwei Gas limit: 12000000 Nonce: 11
ROAStakingFactory.setRewardPool confirmed Block: 13 Gas used: 44622 (0.37%)

tests/test_staking_factory.py::test_set_reward_pool PASSED
```

Annexes

Testing code:

Test Reward Distributor:

```
from brownie import (
    reverts,
)

from scripts.helpful_scripts import (
    ZERO_ADDRESS,
    DAY_TIMESTAMP,
    get_account,
    increase_timestamp
)

from scripts.deploy import (
    deploy_erc,
    deploy_access_control,
    deploy_roai_staking,
    deploy_roai_factory,
    deploy_reward_dist
)

def test_distributeRewards(only_local):
    # Arrange
    owner = get_account(0)
    other = get_account(1)
    extra = get_account(2)
```

```

treasury = get_account(8)

reward_dist = deploy_reward_dist(owner, treasury)

with reverts("Ownable: caller is not the owner"):
    reward_dist.distributeRewards([], [], {"from": other})

with reverts("No pools specified"):
    reward_dist.distributeRewards([], [], {"from": owner})

with reverts("Arrays length mismatch"):
    reward_dist.distributeRewards([extra], [50, 50], {"from": owner})

with reverts("Percentages must sum to 100"):
    reward_dist.distributeRewards([extra], [50], {"from": owner})

with reverts("No funds available for distribution"):
    reward_dist.distributeRewards([extra], [100], {"from": owner})

owner.transfer(reward_dist.address, "1 ether")

with reverts("Invalid pool address"):
    reward_dist.distributeRewards([ZERO_ADDRESS], [100], {"from": owner})

tx = reward_dist.distributeRewards([other], [100], {"from": owner})

assert tx.events['RewardsDistributed'][0]['pools'][0] == other
assert tx.events['RewardsDistributed'][0]['amounts'][0] == 0.8e18

def test_set_reserve_perc(only_local):
    # Arrange

    owner = get_account(0)

    other = get_account(1)

```

```

extra = get_account(2)

treasury = get_account(8)

reward_dist = deploy_reward_dist(owner, treasury)

with reverts("Ownable: caller is not the owner"):
    reward_dist.setReservePercentage(50, {"from": other})

with reverts("Reserve cannot exceed 50%"):
    reward_dist.setReservePercentage(75, {"from": owner})

assert reward_dist.reservePercentage() == 20

reward_dist.setReservePercentage(50, {"from": owner})

assert reward_dist.reservePercentage() == 50

```

```

def test_set_treasury(only_local):

    # Arrange

    owner = get_account(0)

    other = get_account(1)

    new_treasury = get_account(2)

    treasury = get_account(8)

    reward_dist = deploy_reward_dist(owner, treasury)

    with reverts("Ownable: caller is not the owner"):
        reward_dist.setTreasuryWallet(new_treasury, {"from": other})

    with reverts("Invalid treasury address"):
        reward_dist.setTreasuryWallet(ZERO_ADDRESS, {"from": owner})

```

```
assert reward_dist.treasuryWallet() == treasury

reward_dist.setTreasuryWallet(new_treasury, {"from": owner})

assert reward_dist.treasuryWallet() == new_treasury
```

Test Staking Factory:

```
from brownie import (
    reverts,
)

from scripts.helpful_scripts import (
    ZERO_ADDRESS,
    get_account,
)

from scripts.deploy import (
    deploy_erc,
    deploy_access_control,
    deploy_roai_factory
)

def test_create_staking_pool(only_local):
    # Arrange

    owner = get_account(0)
    other = get_account(1)
    extra = get_account(2)

    roai_token = deploy_erc(owner, "Roai", "ROAI")
    access_control = deploy_access_control(owner)
```



```
factory = deploy_roai_factory(owner, roai_token.address)
```

```
with reverts("Invalid max stakers"):
```

```
    factory.createStakingPool(11000, 10, "POOL")
```

```
with reverts("Invalid stake amount"):
```

```
    factory.createStakingPool(1000, 0, "POOL")
```

```
with reverts("Invalid pool name"):
```

```
    factory.createStakingPool(1000, 10, "")
```

```
tx = factory.createStakingPool(1000, 10, "POOL", {"from": owner})
```

```
assert tx.events['StakingPoolCreated'][0]['maxStakers'] == 1000
```

```
assert tx.events['StakingPoolCreated'][0]['fixedStakeAmount'] == 10
```

```
assert tx.events['StakingPoolCreated'][0]['name'] == "POOL"
```

```
def test_set_reward_pool(only_local):
```

```
    # Arrange
```

```
    owner = get_account(0)
```

```
    other = get_account(1)
```

```
    fake_reward_pool = get_account(2)
```

```
    roai_token = deploy_erc(owner, "Roai", "ROAI")
```

```
    access_control = deploy_access_control(owner)
```

```
    factory = deploy_roai_factory(owner, roai_token.address)
```

```
    with reverts("Ownable: caller is not the owner"):
```

```
        factory.setRewardPool(fake_reward_pool, {"from": other})
```

```
    with reverts("Invalid reward pool address"):
```

```

        factory.setRewardPool(ZERO_ADDRESS, {"from": owner})

    assert factory.rewardPool() == ZERO_ADDRESS

    factory.setRewardPool(fake_reward_pool, {"from": owner})

    assert factory.rewardPool() == fake_reward_pool

def test_set_access_control(only_local):

    # Arrange

    owner = get_account(0)

    other = get_account(1)

    roai_token = deploy_erc(owner, "Roai", "ROAI")

    access_control = deploy_access_control(owner)

    factory = deploy_roai_factory(owner, roai_token.address)

    with reverts("Ownable: caller is not the owner"):

        factory.setAccessControl(access_control.address, {"from": other})

    with reverts("Invalid AccessControl address"):

        factory.setAccessControl(ZERO_ADDRESS, {"from": owner})

    assert factory.accessControl() == ZERO_ADDRESS

    factory.setAccessControl(access_control.address, {"from": owner})

    assert factory.accessControl() == access_control.address

def test_distribute_rewards(only_local):

    # Arrange

    owner = get_account(0)

    other = get_account(1)

```

```

fake_reward_pool = get_account(2)

roai_token = deploy_erc(owner, "Roai", "ROAI")

factory = deploy_roai_factory(owner, roai_token.address)
factory.setRewardPool(fake_reward_pool, {"from": owner})

with reverts("Only reward pool can distribute"):
    factory.distributeRewardsToMultiplePools(
        [], [], 1e18, {"from": owner}
    )

with reverts("Percentages must sum to 100"):
    factory.distributeRewardsToMultiplePools(
        [], [], 1e18, {"from": fake_reward_pool}
    )

with reverts("Arrays must be same length"):
    factory.distributeRewardsToMultiplePools(
        [], [50], 1e18, {"from": fake_reward_pool}
    )

with reverts("Not a valid staking pool"):
    factory.distributeRewardsToMultiplePools(
        [other], [100], 1e18, {"from": fake_reward_pool}
    )

tx = factory.createStakingPool(1000, 10, "POOL", {"from": owner})
pool_addr = tx.events['StakingPoolCreated'][0]['poolAddress']

```

```

# no eth in contract

with reverts("ETH transfer failed to pool"):

    factory.distributeRewardsToMultiplePools(

        [pool_addr], [100], 1e18, {"from": fake_reward_pool}

    )

owner.transfer(factory.address, "1 ether")

tx = factory.distributeRewardsToMultiplePools(

    [pool_addr], [100], 1e18, {"from": fake_reward_pool}

)

assert tx.events['RewardsAdded'][0]['amount'] == 1e18

assert tx.events['RewardsDistributed'][0]['pools'][0] == pool_addr

assert tx.events['RewardsDistributed'][0]['percentages'][0] == 100

```

Test Staking Pool:

```

from brownie import (

    reverts,

)

from scripts.helpful_scripts import (

    ZERO_ADDRESS,

    DAY_TIMESTAMP,

    get_account,

    get_timestamp,

    get_chain_number,

    increase_timestamp

)

```

```

from scripts.deploy import (
    deploy_erc,
    deploy_access_control,
    deploy_roai_staking
)

def test_constructor(only_local):
    # Arrange
    owner = get_account(0)
    other = get_account(1)
    extra = get_account(2)

    roai_token = deploy_erc(owner, "Roai", "ROAI")
    access_control = deploy_access_control(owner)

    with reverts("Invalid token address"):
        deploy_roai_staking(
            owner, ZERO_ADDRESS, 15,
            50000, access_control.address)

    with reverts("Invalid max stakers"):
        deploy_roai_staking(
            owner, roai_token.address, 10001,
            50000, access_control.address)

    with reverts("Invalid stake amount"):
        deploy_roai_staking(
            owner, roai_token.address, 1000,
            0, access_control.address)

```



```

staking = deploy_roai_staking(
    owner, roai_token.address, 15,
    50000, access_control.address)

def test_stake(only_local):
    # Arrange

    owner = get_account(0)
    other = get_account(1)
    extra = get_account(2)
    another = get_account(3)

    roai_token = deploy_erc(owner, "Roai", "ROAI")
    access_control = deploy_access_control(owner)
    staking = deploy_roai_staking(
        owner, roai_token.address, 2,
        10, access_control.address)

    ...

    _addLockPeriod(0, 0, true); // 1 minute for testing (0 days + 1 minute in seconds)
    _addLockPeriod(30, 5, true); // 30 days, 5% bonus
    _addLockPeriod(60, 15, true); // 60 days, 15% bonus
    _addLockPeriod(90, 25, true); // 90 days, 25% bonus
    _addLockPeriod(180, 35, true); // 180 days, 35% bonus
    ...

    with reverts("Invalid lock period index"):
        staking.stake(5, {"from": other})

```

```

with reverts("Access denied: Pool Manager role required"):

    staking.disableLockPeriod(0, {"from": other})

staking.disableLockPeriod(0, {"from": owner})

with reverts("Lock period not active"):

    staking.stake(0, {"from": other})


# mint some tokens

roai_token.mint(other, 50e9)

roai_token.mint(extra, 50e9)

roai_token.mint(another, 50e9)

roai_token.approve(staking.address, 1000e9, {"from": other})

roai_token.approve(staking.address, 1000e9, {"from": extra})

roai_token.approve(staking.address, 1000e9, {"from": another})


tx = staking.stake(1, {"from": other})

assert tx.events['Staked'][0]['user'] == other

assert tx.events['Staked'][0]['amount'] == 10e9

assert tx.events['Staked'][0]['lockPeriodIndex'] == 1

assert tx.events['Transfer'][0]['from'] == other

assert tx.events['Transfer'][0]['to'] == staking.address

assert tx.events['Transfer'][0]['value'] == 10e9


with reverts("User has already staked in this pool"):

    staking.stake(1, {"from": other})


tx = staking.stake(1, {"from": extra})

assert tx.events['Staked'][0]['user'] == extra

assert tx.events['Staked'][0]['amount'] == 10e9

```

```
assert tx.events['Staked'][0]['lockPeriodIndex'] == 1

assert tx.events['Transfer'][0]['from'] == extra

assert tx.events['Transfer'][0]['to'] == staking.address

assert tx.events['Transfer'][0]['value'] == 10e9
```

```
with reverts("Pool is full"):

    staking.stake(1, {"from": another})
```

```
def test_ustake(only_local):
```

```
    # Arrange
```

```
    owner = get_account(0)
```

```
    other = get_account(1)
```

```
    extra = get_account(2)
```

```
    roai_token = deploy_erc(owner, "Roai", "ROAI")
```

```
    access_control = deploy_access_control(owner)
```

```
    staking = deploy_roai_staking(

        owner, roai_token.address, 5,

        10, access_control.address)
```

```
    # mint some tokens
```

```
    roai_token.mint(other, 50e9)
```

```
    roai_token.mint(extra, 50e9)
```

```
    roai_token.approve(staking.address, 1000e9, {"from": other})
```

```
    roai_token.approve(staking.address, 1000e9, {"from": extra})
```

```
    staking.stake(1, {"from": other})
```

```

staking.stake(2, {"from": extra})

with reverts("Invalid position index"):
    staking.unstake(2, {"from": other})

with reverts("Lock period not ended"):
    staking.unstake(0, {"from": other})

owner.transfer(staking.address, "1 ether")

increase_timestamp(DAY_TIMESTAMP * 35)

tx = staking.unstake(0, {"from": other})

assert tx.events['Transfer'][0]['from'] == staking.address
assert tx.events['Transfer'][0]['to'] == other
assert tx.events['Transfer'][0]['value'] == 10e9

```

```

def test_claim_rewards(only_local):

    # Arrange

    owner = get_account(0)
    other = get_account(1)
    extra = get_account(2)
    another = get_account(3)

    roai_token = deploy_erc(owner, "Roai", "ROAI")
    access_control = deploy_access_control(owner)
    staking = deploy_roai_staking(
        owner, roai_token.address, 5,
        10, access_control.address)

    # mint some tokens

```

```
roai_token.mint(other, 50e9)

roai_token.mint(extra, 50e9)

roai_token.mint(another, 50e9)

roai_token.approve(staking.address, 1000e9, {"from": other})
roai_token.approve(staking.address, 1000e9, {"from": extra})
roai_token.approve(staking.address, 1000e9, {"from": another})


# some stakes

staking.stake(1, {"from": other})

staking.stake(2, {"from": extra})

staking.stake(3, {"from": another})


owner.transfer(staking.address, "1 ether")


with reverts("Invalid position index"):

    staking.claimRewards(1, {"from": other})

with reverts("No rewards to claim"):

    staking.claimRewards(0, {"from": other})

increase_timestamp(DAY_TIMESTAMP * 35)


tx = staking.claimRewards(0, {"from": other})

assert tx.events['RewardsClaimed'][0]['user'] == other

assert tx.events['RewardsClaimed'][0]['positionIndex'] == 0

assert staking.calculateRewards(other, 0) == 0


increase_timestamp(DAY_TIMESTAMP * 35)


tx = staking.claimRewards(0, {"from": extra})
```



```
assert tx.events['RewardsClaimed'][0]['user'] == extra

assert tx.events['RewardsClaimed'][0]['positionIndex'] == 0

assert staking.calculateRewards(extra, 0) == 0
```

Technical Findings Summary

Findings

Vulnerability Level	Total	Pending	Not Apply	Acknowledged	Partially Fixed	Fixed
<div><div></div>High</div>	2					2
<div><div></div>Medium</div>	6			1	1	4
<div><div></div>Low</div>	1					1
<div><div></div>Informational</div>	2			2		

Assessment Results

Score Results

Review	Score
Global Score	95/100
Assure KYC	https://projects.assuredefi.com/project/roai-scalper
Audit Score	95/100

The Following Score System Has been Added to this page to help understand the value of the audit, the maximum score is 100, however to attain that value the project must pass and provide all the data needed for the assessment. Our Passing Score has been changed to 84 Points for a higher standard, if a project does not attain 85% is an automatic failure. Read our notes and final assessment below. The Global Score is a combination of the evaluations obtained between having or not having KYC and the type of contract audited together with its manual audit.

Audit PASS

Following our comprehensive security audit of the token contract for the ROAI project, we inform you that the project has met the necessary security standards.

Disclaimer

Assure Defi has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocating for the Project, and users relying on this report should not consider this as having any merit for financial adROAI in any shape, form, or nature. The contracts audited do not account for any economic developments that the Project in question may pursue, and the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude, and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are entirely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment adROAI, nor should it be used to signal that any person reading this report should invest their funds without sufficient individual due diligence, regardless of the findings presented. Information is provided 'as is, and Assure Defi is under no covenant to audit completeness, accuracy, or solidity of the contracts. In no event will Assure Defi or its partners, employees, agents, or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions or actions with regards to the information provided in this audit report.

The assessment serROAI provided by Assure Defi are subject to dependencies and are under continuing development. You agree that your access or use, including but not limited to any serROAI, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The serROAI may access, and depend upon, multiple layers of third parties.