Assure DeFi®

THE VERIFICATION GOLD STANDARD

VERIFIED BY ASSURE DEFI ®
INTEGRITY ✦ TRUST ✦ CREDIBILITY

Security Assessment

Sorra Staking

Date: 18/12/2024

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🟡 Low	Low-level vulnerabilities are primarily associated with outdated or unused code snippets that generally do not significantly impact execution, sometimes they can be ignored.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 **Well Secured.**

Insecure	**Poorly Secured**	**Secured**	**Well Secured**

Scope

Target Code And Revision

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

Target Code And Revision

Project	Assure
Language	Solidity
Codebase	sorraStaking.sol [SHA256]: 7111e2f6075d026dc9ae78719bc04c959e5bfe579102e5baba4c352c34623baa sorraStaking.sol - Fixed version [SHA256]: 53af1c33d8affc66a1270780a08b362c2e888c0e4a8a2313b5308f61242cdabe
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	● 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

HIGH

1. Incorrect Index Handling After pop() in _decreasePosition() Function [Fixed ✅]

Function: _decreasePosition()

Issue: There is an incorrect handling of the index i in the for loop after performing a pop() on the position.deposits array. When pop() is called, it removes the element correctly, but then the index i is decremented with i--. This causes the index i to revert to the previous position, leading to the same index being processed in the next iteration, which could result in an infinite loop. Additionally, this bug causes the user to lose rewards.

Recommendation: Modify the for loop to avoid directly relying on the index i, and instead, implement a safer method to iterate over the array when elements are removed.

Update: For loop was modified and now the interaction is safe.

MEDIUM

No medium severity issues were found.

LOW

No low severity issues were found.

INFORMATIONAL

1. Unused uint256 position in _calculateRewards() Function [Fixed ✅]

Function: _calculateRewards()

Issue: The uint256 position parameter is not being utilized within the function. This results in unnecessary code and potential confusion, as the parameter is passed into the function but never referenced or used in any part of the logic.

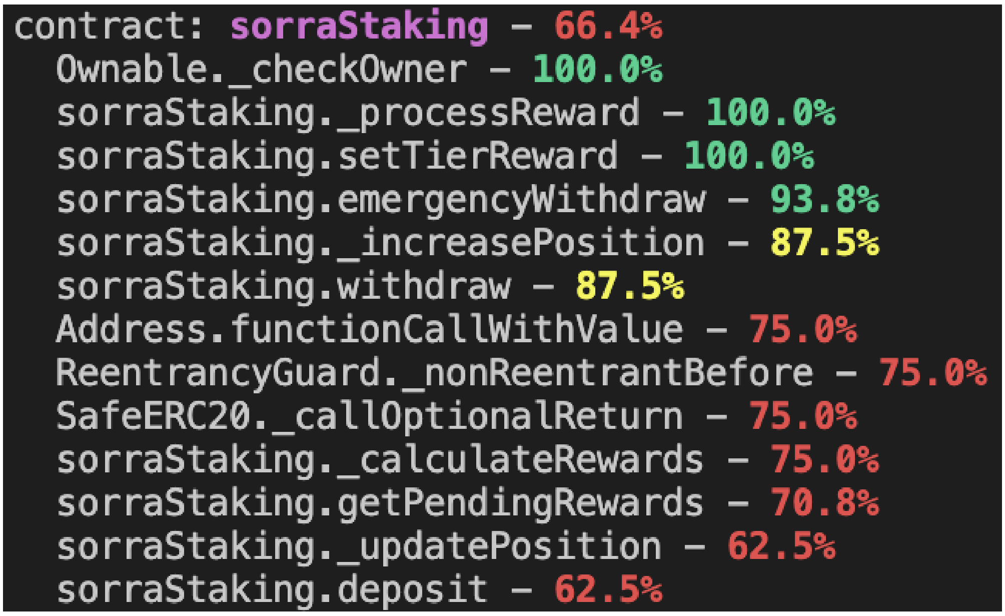
Recommendation: Remove the unused uint256 position parameter from the function if it is not needed, or implement its usage within the function logic if it is intended to be used.

Update: Unused variables were removed.

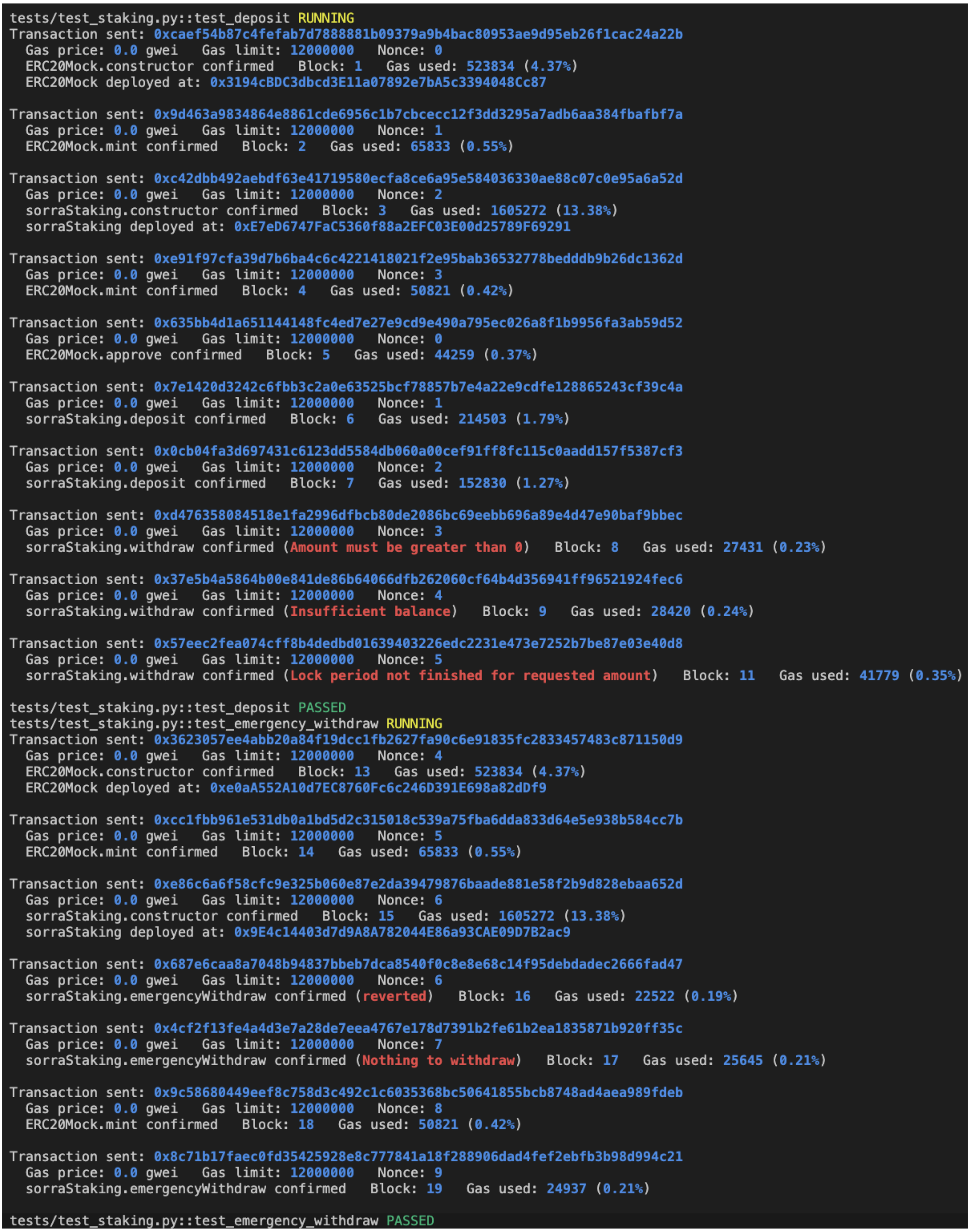
Testing coverage

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

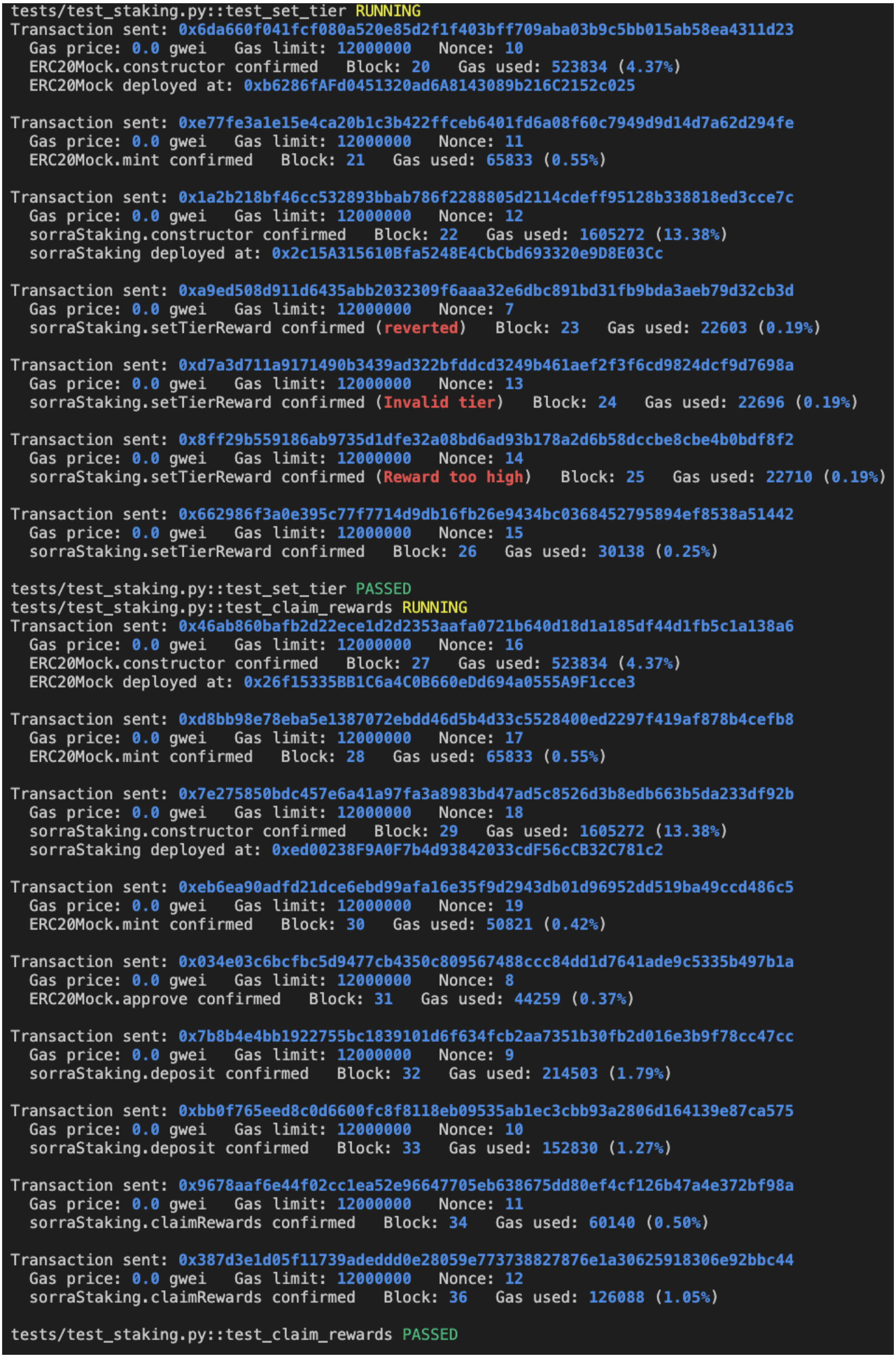
Sorra Staking contract tests:



```
contract: sorraStaking - 66.4%
  Ownable._checkOwner - 100.0%
  sorraStaking._processReward - 100.0%
  sorraStaking.setTierReward - 100.0%
  sorraStaking.emergencyWithdraw - 93.8%
  sorraStaking._increasePosition - 87.5%
  sorraStaking.withdraw - 87.5%
  Address.functionCallWithValue - 75.0%
  ReentrancyGuard._nonReentrantBefore - 75.0%
  SafeERC20._callOptionalReturn - 75.0%
  sorraStaking._calculateRewards - 75.0%
  sorraStaking.getPendingRewards - 70.8%
  sorraStaking._updatePosition - 62.5%
  sorraStaking.deposit - 62.5%
```



```
tests/test_staking.py::test_deposit RUNNING
Transaction sent: 0xcaef54b87c4fefab7d7888881b09379a9b4bac80953ae9d95eb26f1cac24a22b
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  ERC20Mock.constructor confirmed   Block: 1   Gas used: 523834 (4.37%)
  ERC20Mock deployed at: 0x3194cBDC3dbcd3E11a07892e7bA5c3394048Cc87

Transaction sent: 0x9d463a9834864e8861cde6956c1b7cbcecc12f3dd3295a7adb6aa384fbafbf7a
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 1
  ERC20Mock.mint confirmed   Block: 2   Gas used: 65833 (0.55%)

Transaction sent: 0xc42dbb492aebdf63e41719580ecfa8ce6a95e584036330ae88c07c0e95a6a52d
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 2
  sorraStaking.constructor confirmed   Block: 3   Gas used: 1605272 (13.38%)
  sorraStaking deployed at: 0xE7eD6747FaC5360f88a2EFC03E00d25789F69291

Transaction sent: 0xe91f97cfa39d7b6ba4c6c4221418021f2e95bab36532778bedddb9b26dc1362d
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 3
  ERC20Mock.mint confirmed   Block: 4   Gas used: 50821 (0.42%)

Transaction sent: 0x635bb4d1a651144148fc4ed7e27e9cd9e490a795ec026a8f1b9956fa3ab59d52
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  ERC20Mock.approve confirmed   Block: 5   Gas used: 44259 (0.37%)

Transaction sent: 0x7e1420d3242c6fbb3c2a0e63525bcf78857b7e4a22e9cdfe128865243cf39c4a
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 1
  sorraStaking.deposit confirmed   Block: 6   Gas used: 214503 (1.79%)

Transaction sent: 0x0cb04fa3d697431c6123dd5584db060a00cef91ff8fc115c0aadd157f5387cf3
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 2
  sorraStaking.deposit confirmed   Block: 7   Gas used: 152830 (1.27%)

Transaction sent: 0xd476358084518e1fa2996dfbcb80de2086bc69eebb696a89e4d47e90baf9bbec
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 3
  sorraStaking.withdraw confirmed (Amount must be greater than 0)   Block: 8   Gas used: 27431 (0.23%)

Transaction sent: 0x37e5b4a5864b00e841de86b64066dfb262060cf64b4d356941ff96521924fec6
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 4
  sorraStaking.withdraw confirmed (Insufficient balance)   Block: 9   Gas used: 28420 (0.24%)

Transaction sent: 0x57eec2fea074cff8b4dedbd01639403226edc2231e473e7252b7be87e03e40d8
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 5
  sorraStaking.withdraw confirmed (Lock period not finished for requested amount)   Block: 11   Gas used: 41779 (0.35%)

tests/test_staking.py::test_deposit PASSED
tests/test_staking.py::test_emergency_withdraw RUNNING
Transaction sent: 0x3623057ee4abb20a84f19dcc1fb2627fa90c6e91835fc2833457483c871150d9
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 4
  ERC20Mock.constructor confirmed   Block: 13   Gas used: 523834 (4.37%)
  ERC20Mock deployed at: 0xe0aA552A10d7EC8760Fc6c246D391E698a82dDf9

Transaction sent: 0xcc1fbb961e531db0a1bd5d2c315018c539a75fba6dda833d64e5e938b584cc7b
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 5
  ERC20Mock.mint confirmed   Block: 14   Gas used: 65833 (0.55%)

Transaction sent: 0xe86c6a6f58cfc9e325b060e87e2da39479876baade881e58f2b9d828ebaa652d
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 6
  sorraStaking.constructor confirmed   Block: 15   Gas used: 1605272 (13.38%)
  sorraStaking deployed at: 0x9E4c14403d7d9A8A782044E86a93CAE09D7B2ac9

Transaction sent: 0x687e6caa8a7048b94837bbeb7dca8540f0c8e8e68c14f95debdadec2666fad47
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 6
  sorraStaking.emergencyWithdraw confirmed (reverted)   Block: 16   Gas used: 22522 (0.19%)

Transaction sent: 0x4cf2f13fe4a4d3e7a28de7eea4767e178d7391b2fe61b2ea1835871b920ff35c
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 7
  sorraStaking.emergencyWithdraw confirmed (Nothing to withdraw)   Block: 17   Gas used: 25645 (0.21%)

Transaction sent: 0x9c58680449eef8c758d3c492c1c6035368bc50641855bcb8748ad4aea989fdeb
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 8
  ERC20Mock.mint confirmed   Block: 18   Gas used: 50821 (0.42%)

Transaction sent: 0x8c71b17faec0fd35425928e8c777841a18f288906dad4fef2ebfb3b98d994c21
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 9
  sorraStaking.emergencyWithdraw confirmed   Block: 19   Gas used: 24937 (0.21%)

tests/test_staking.py::test_emergency_withdraw PASSED
```



```
tests/test_staking.py::test_set_tier RUNNING
Transaction sent: 0x6da660f041fcf080a520e85d2f1f403bff709aba03b9c5bb015ab58ea4311d23
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 10
  ERC20Mock.constructor confirmed   Block: 20   Gas used: 523834 (4.37%)
  ERC20Mock deployed at: 0xb6286fAFd0451320ad6A8143089b216C2152c025

Transaction sent: 0xe77fe3a1e15e4ca20b1c3b422ffceb6401fd6a08f60c7949d9d14d7a62d294fe
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 11
  ERC20Mock.mint confirmed   Block: 21   Gas used: 65833 (0.55%)

Transaction sent: 0x1a2b218bf46cc532893bbab786f2288805d2114cdeff95128b338818ed3cce7c
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 12
  sorraStaking.constructor confirmed   Block: 22   Gas used: 1605272 (13.38%)
  sorraStaking deployed at: 0x2c15A315610Bfa5248E4CbCbd693320e9D8E03Cc

Transaction sent: 0xa9ed508d911d6435abb2032309f6aaa32e6dbc891bd31fb9bda3aeb79d32cb3d
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 7
  sorraStaking.setTierReward confirmed (reverted)   Block: 23   Gas used: 22603 (0.19%)

Transaction sent: 0xd7a3d711a9171490b3439ad322bfddcd3249b461aef2f3f6cd9824dcf9d7698a
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 13
  sorraStaking.setTierReward confirmed (Invalid tier)   Block: 24   Gas used: 22696 (0.19%)

Transaction sent: 0x8ff29b559186ab9735d1dfe32a08bd6ad93b178a2d6b58dccbe8cbe4b0bdf8f2
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 14
  sorraStaking.setTierReward confirmed (Reward too high)   Block: 25   Gas used: 22710 (0.19%)

Transaction sent: 0x662986f3a0e395c77f7714d9db16fb26e9434bc0368452795894ef8538a51442
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 15
  sorraStaking.setTierReward confirmed   Block: 26   Gas used: 30138 (0.25%)

tests/test_staking.py::test_set_tier PASSED
tests/test_staking.py::test_claim_rewards RUNNING
Transaction sent: 0x46ab860bafb2d22ece1d2d2353aafa0721b640d18d1a185df44d1fb5c1a138a6
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 16
  ERC20Mock.constructor confirmed   Block: 27   Gas used: 523834 (4.37%)
  ERC20Mock deployed at: 0x26f15335BB1C6a4C0B660eDd694a0555A9F1cce3

Transaction sent: 0xd8bb98e78eba5e1387072ebdd46d5b4d33c5528400ed2297f419af878b4cefb8
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 17
  ERC20Mock.mint confirmed   Block: 28   Gas used: 65833 (0.55%)

Transaction sent: 0x7e275850bdc457e6a41a97fa3a8983bd47ad5c8526d3b8edb663b5da233df92b
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 18
  sorraStaking.constructor confirmed   Block: 29   Gas used: 1605272 (13.38%)
  sorraStaking deployed at: 0xed00238F9A0F7b4d93842033cdF56cCB32C781c2

Transaction sent: 0xeb6ea90adfd21dce6ebd99afa16e35f9d2943db01d96952dd519ba49ccd486c5
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 19
  ERC20Mock.mint confirmed   Block: 30   Gas used: 50821 (0.42%)

Transaction sent: 0x034e03c6bcfbc5d9477cb4350c809567488ccc84dd1d7641ade9c5335b497b1a
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 8
  ERC20Mock.approve confirmed   Block: 31   Gas used: 44259 (0.37%)

Transaction sent: 0x7b8b4e4bb1922755bc1839101d6f634fcb2aa7351b30fb2d016e3b9f78cc47cc
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 9
  sorraStaking.deposit confirmed   Block: 32   Gas used: 214503 (1.79%)

Transaction sent: 0xbb0f765eed8c0d6600fc8f8118eb09535ab1ec3cbb93a2806d164139e87ca575
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 10
  sorraStaking.deposit confirmed   Block: 33   Gas used: 152830 (1.27%)

Transaction sent: 0x9678aaf6e44f02cc1ea52e96647705eb638675dd80ef4cf126b47a4e372bf98a
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 11
  sorraStaking.claimRewards confirmed   Block: 34   Gas used: 60140 (0.50%)

Transaction sent: 0x387d3e1d05f11739adeddd0e28059e773738827876e1a30625918306e92bbc44
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 12
  sorraStaking.claimRewards confirmed   Block: 36   Gas used: 126088 (1.05%)

tests/test_staking.py::test_claim_rewards PASSED
```

Annexes

Testing code:

Test sorra Staking contract:

```python
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_staking
)

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

    reward_token = deploy_erc(owner, "Reward", "RWD")

    with reverts("Zero address"):
        deploy_staking(owner, ZERO_ADDRESS)
    with reverts("Invalid token"):
        deploy_staking(owner, reward_token.address)

    reward_token.mint(owner, 100e18)
    staking = deploy_staking(owner, reward_token.address)

    with reverts():
        staking.setDepositingEnabled(False, {"from": other})
    tx = staking.setDepositingEnabled(False, {"from": owner})
    assert tx.events['DepositingStatusChanged'][0]['enabled'] == False

    with reverts("Deposits are disabled"):
        staking.deposit(0.5e18, 5, {"from": other})
    staking.setDepositingEnabled(True, {"from": owner})
    with reverts("Invalid tier"):
        staking.deposit(0.5e18, 5, {"from": other})

    reward_token.mint(other, 1e18)
    reward_token.approve(staking.address, 1e18, {"from": other})
    tx = staking.deposit(0.5e18, 0, {"from": other})
    assert tx.events['Transfer'][0]['from'] == other
    assert tx.events['Transfer'][0]['to'] == staking.address
    assert tx.events['Transfer'][0]['value'] == 0.5e18
    assert tx.events['Depositx'][0]['user'] == other
```

```python
    assert tx.events['Depositx'][0]['amount'] == 0.5e18

    # try to mint more than 5 deposits

    tx = staking.deposit(0.5e18, 0, {"from": other})
    assert tx.events['Transfer'][0]['from'] == other
    assert tx.events['Transfer'][0]['to'] == staking.address
    assert tx.events['Transfer'][0]['value'] == 0.5e18

    reward_token.mint(other, 5e18)
    reward_token.approve(staking.address, 5e18, {"from": other})
    staking.deposit(0.5e18, 0, {"from": other})
    staking.deposit(0.5e18, 0, {"from": other})
    staking.deposit(0.5e18, 0, {"from": other})

    with reverts("Too many deposits"):
        staking.deposit(0.5e18, 0, {"from": other})

    increase_timestamp(DAY_TIMESTAMP * 65)
    assert staking.getPendingRewards(other) == 0.125e18

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

    reward_token = deploy_erc(owner, "Reward", "RWD")
    reward_token.mint(owner, 100e18)
    staking = deploy_staking(owner, reward_token.address)

    # Mint some tokens
    reward_token.mint(other, 2e18)
    reward_token.approve(staking.address, 2e18, {"from": other})

    # Deposit some tokens
    staking.deposit(0.5e18, 0, {"from": other})
    staking.deposit(0.5e18, 1, {"from": other})
    with reverts("Amount must be greater than 0"):
        staking.withdraw(0, {"from": other})
    with reverts("Insufficient balance"):
        staking.withdraw(2e18, {"from": other})

    increase_timestamp(DAY_TIMESTAMP * 1)

    with reverts("Lock period not finished for requested amount"):
        staking.withdraw(0.5e18, {"from": other})

    increase_timestamp(DAY_TIMESTAMP * 15)
    # TODO; Fix
    #staking.withdraw(0.5e18, {"from": other})

def test_emergency_withdraw(only_local):
    # Arrange
    owner = get_account(0)
    other = get_account(1)
    reward_token = deploy_erc(owner, "Reward", "RWD")
    reward_token.mint(owner, 100e18)
    staking = deploy_staking(owner, reward_token.address)
    with reverts():
        staking.emergencyWithdraw(1e18, {"from": other})
    with reverts("Nothing to withdraw"):
        staking.emergencyWithdraw(0, {"from": owner})

    reward_token.mint(staking.address, 1e18)
    tx = staking.emergencyWithdraw(1e18, {"from": owner})
    assert tx.events['Transfer'][0]['from'] == staking.address
    assert tx.events['Transfer'][0]['to'] == owner
    assert tx.events['Transfer'][0]['value'] == 1e18

def test_set_tier(only_local):
    # Arrange
    owner = get_account(0)
    other = get_account(1)

    reward_token = deploy_erc(owner, "Reward", "RWD")
    reward_token.mint(owner, 100e18)
    staking = deploy_staking(owner, reward_token.address)
    with reverts():
        staking.setTierReward(0, 1000, {"from": other})
    with reverts("Invalid tier"):
        staking.setTierReward(4, 1000, {"from": owner})
    with reverts("Reward too high"):
        staking.setTierReward(0, 20000, {"from": owner})

    tx = staking.setTierReward(0, 1000, {"from": owner})
    assert tx.events['RewardBpsUpdated'][0]['tier'] == 0
    assert tx.events['RewardBpsUpdated'][0]['oldBps'] == 500
    assert tx.events['RewardBpsUpdated'][0]['newBps'] == 1000

def test_claim_rewards(only_local):
    # Arrange
    owner = get_account(0)
    other = get_account(1)

    reward_token = deploy_erc(owner, "Reward", "RWD")
    reward_token.mint(owner, 100e18)
    staking = deploy_staking(owner, reward_token.address)

    # Mint some tokens
    reward_token.mint(other, 2e18)
    reward_token.approve(staking.address, 2e18, {"from": other})

    # Deposit some tokens
    staking.deposit(0.5e18, 0, {"from": other})
    staking.deposit(0.5e18, 1, {"from": other})

    assert staking.getPendingRewards(other) == 0
    staking.claimRewards({"from": other})

    increase_timestamp(DAY_TIMESTAMP * 16)
    assert staking.getPendingRewards(other) == 0.025e18


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

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

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

    assert tx.events['Transfer'][0]['value'] == 0.025e18

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

    assert tx.events['RewardDistributed'][0]['amount'] == 0.025e18
```

Technical Findings Summary

Findings

Vulnerability Level	Total	Pending	Not Apply	Acknowledged	Partially Fixed	Fixed
🔴 High	1					1
🟠 Medium	0					
🟡 Low	0					
🟢 Informational	1					1

Assessment Results

Score Results

Review	Score
Global Score	**95/100**
Assure KYC	https://assuredefi.com/projects/sorra
Audit Score	90/100

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 Sorra Staking project, we inform you that the contract has not met the required standards and a solution must be implemented for the high vulnerability detected. Update: All the issues were fixed by the development team.

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 advice 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 advice, 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 services 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 services, 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 services may access, and depend upon, multiple layers of third parties.