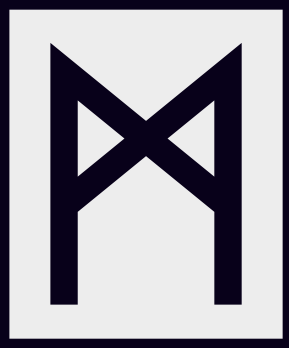


AUKI

SMART CONTRACTS REVIEW

zokyo

October 30th 2024 | v. 1.0

Security Audit Score

PASS

Zokyo Security has concluded that this smart contract passed a security audit.

ZOKYO AUDIT SCORING AUKI LABS

1. Severity of Issues:
    - Critical: Direct, immediate risks to funds or the integrity of the contract. Typically, these would have a very high weight.
    - High: Important issues that can compromise the contract in certain scenarios.
    - Medium: Issues that might not pose immediate threats but represent significant deviations from best practices.
    - Low: Smaller issues that might not pose security risks but are still noteworthy.
    - Informational: Generally, observations or suggestions that don't point to vulnerabilities but can be improvements or best practices.
2. Test Coverage: The percentage of the codebase that's covered by tests. High test coverage often suggests thorough testing practices and can increase the score.
3. Code Quality: This is more subjective, but contracts that follow best practices, are well-commented, and show good organization might receive higher scores.
4. Documentation: Comprehensive and clear documentation might improve the score, as it shows thoroughness.
5. Consistency: Consistency in coding patterns, naming, etc., can also factor into the score.
6. Response to Identified Issues: Some audits might consider how quickly and effectively the team responds to identified issues.

SCORING CALCULATION:

Let's assume each issue has a weight:
- Critical: -30 points
- High: -20 points
- Medium: -10 points
- Low: -5 points
- Informational: -1 point

Starting with a perfect score of 100:
- 0 Critical issues: 0 points deducted
- 0 High issues: 0 points deducted
- 0 Medium issues: 0 points deducted
- 1 Low issue: 1 resolved = 0 points deducted
- 5 Informational issues: 5 resolved = 0 points deducted

Thus, the scope is 100

TECHNICAL SUMMARY

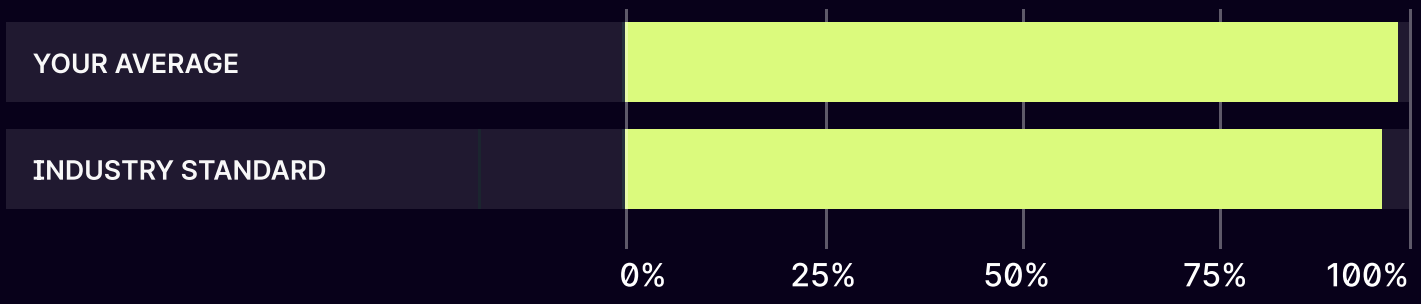
This document outlines the overall security of the Auki Labs smart contract/s evaluated by the Zokyo Security team.

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

Contract Status

There were 0 critical issues found during the review. (See Complete Analysis)

Testable Code



100% of the code is testable, 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/s but rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that can withstand the Ethereum network's fast-paced and rapidly changing environment, we recommend that the Auki Labs team put in place a bug bounty program to encourage further active analysis of the smart contract/s.

Table of Contents

AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contract was taken from the Auki Labs repository:
Repo: https://github.com/aukilabs/auki-contracts/pull/72

Last commit - f87591239e2170b12ad9e3f3fad9d61abaf94dc2

Within the scope of this audit, the team of auditors reviewed the following contract(s):

- ./DomainNFT.sol
- ./StakingWithDelayedWithdrawalContract.sol
- ./RewardLiquidityPoolContract.sol
- ./UUPSProxy.sol
- ./StakingERC1155Contract.sol
- ./StakingContract.sol
- ./AukiToken.sol
- ./BurnContract.sol

During the audit, Zokyo Security ensured that the contract:

- Implements and adheres to the existing standards appropriately and effectively;
- The documentation and code comments match the logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices, efficiently using resources without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the most recent vulnerabilities;
- Meets best practices in code readability, etc.

Zokyo Security has followed best practices and industry-standard techniques to verify the implementation of Auki Labs smart contract/s. To do so, the code was reviewed line by line by our smart contract/s developers, who documented even minor issues as they were discovered. Part of this work includes writing a test suite using the Foundry testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

01	Due diligence in assessing the overall code quality of the codebase.	03	Testing contract/s logic against common and uncommon attack vectors.
02	Cross-comparison with other, similar smart contract/s by industry leaders.	04	Thorough manual review of the codebase line by line.

Executive Summary

The Zokyo team has performed a security audit of the provided codebase. The contracts submitted for auditing are well-crafted and organized. Detailed findings from the audit process are outlined in the "Complete Analysis" section.

Auki network is a decentralized machine perception network and collaborative spatial computing protocol, designed to allow digital devices to securely and privately exchange spatial data and computing power to form a shared understanding of the physical world. Auki tokens are used as the depositing token in the staking contracts and an ERC1155 NFT as the staking token.

STRUCTURE AND ORGANIZATION OF THE DOCUMENT

For the ease of navigation, the following sections are arranged from the most to the least critical ones. Issues are tagged as “Resolved” or “Unresolved” or “Acknowledged” depending on whether they have been fixed or addressed. Acknowledged means that the issue was sent to the Auki Labs team and the Auki Labs team is aware of it, but they have chosen to not solve it. The issues that are tagged as “Verified” contain unclear or suspicious functionality that either needs explanation from the Client or remains disregarded by the Client. 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 contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

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

FINDINGS SUMMARY

#	Title	Risk	Status
1	Lack of emitting an event	Low	Resolved
2	Could use delete keyword	Informational	Resolved
3	Floating pragma	Informational	Resolved
4	Excessive external call to this.nonPendingBalance()	Informational	Resolved
5	Lack of using cached variable	Informational	Resolved
6	initSupplySubTargetSupply Can Be Stored In Variable Instead	Informational	Resolved

LOW-1 | RESOLVED

Lack of emitting an event

Location - DomainNFT.sol

The setStakingContract() is missing an event when the stakingContract variable is updated.

Recommendation:
Add a relative event in the function.

INFORMATIONAL-1 | RESOLVED

Could use delete keyword

Location - StakingContract.sol, StakingERC1155Contract.sol

In the StakingContract.sol, the slash() and _withdraw() functions update all elements of the stakes[_staker] to `0`.
Basically, it could use delete keyword to delete the entire struct.

The same for the slash() and withdraw() functions for the StakingERC1155Contract.sol.

Recommendation:
Please use “delete stakes[_staker];” instead of updating all elements to `0`.

INFORMATIONAL-2 | RESOLVED

Floating pragma

Location - All contracts

All the contracts use pragma solidity `^0.8.20`.
This may result in the contracts being deployed using the wrong pragma version, which is different from the one they were tested with.

Recommendation:
It is recommended to lock the pragma version.

INFORMATIONAL-3 | RESOLVED

Excessive external call to this.nonPendingBalance()

Location - RewardLiquidityPoolContract.sol

The increaseFixedAmount() and increaseDynamicAmount() functions make excessive external calls to this.nonPendingBalance() even if the nonPendingBalance() is not external and it doesn't validate msg.sender.

Recommendation:
Remove this keyword.

INFORMATIONAL-4 | RESOLVED

Lack of using cached variable

Location - StakingERC1155Contract.sol

The withdraw() and slash() functions read _s.amount from storage once more for the zero comparison even if the _amount memory variable is declared.

Recommendation:
Use _amount instead of _s.amount for the zero comparison.

INFORMATIONAL-5 | RESOLVED

initSupplySubTargetSupply Can Be Stored In Variable Instead

In contracts BurnContract.sol and DomainNFT.sol 's calculateMintAmounts() function the initSupplySubTargetSupply is calculated by subtracting the TARGET_TOTAL_SUPPLY from the INITIAL_TOTAL_SUPPLY , since these are constants, the the value of initSupplySubTargetSupply will always be the same , therefore instead of calculating it everytime the function is invoked the value can be stored in a variable instead.

Recommendation:
initSupplySubTargetSupply can be stored in a variable and queried everytime calculateMintAmounts() is invoked.

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

	./StakingERC1155Contract.sol ./StakingContract.sol ./AukiToken.sol ./BurnContract.sol
Re-entrancy	Pass
Access Management Hierarchy	Pass
Arithmetic Over/Under Flows	Pass
Unexpected Ether	Pass
Delegatecall	Pass
Default Public Visibility	Pass
Hidden Malicious Code	Pass
Entropy Illusion (Lack of Randomness)	Pass
External Contract Referencing	Pass
Short Address/ Parameter Attack	Pass
Unchecked CALL Return Values	Pass
Race Conditions / Front Running	Pass
General Denial Of Service (DOS)	Pass
Uninitialized Storage Pointers	Pass
Floating Points and Precision	Pass
Tx.Origin Authentication	Pass
Signatures Replay	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Security

As a part of our work assisting Auki Labs in verifying the correctness of their contract/s code, our team was responsible for writing integration tests using the Foundry testing framework.

The tests were based on the functionality of the code, as well as a review of the Auki Labs contract/s requirements for details about issuance amounts and how the system handles these.

Ran 29 tests for test/AukiToken.t.sol:AukiTokenTest
[PASS] testApproveToZeroAddress() (gas: 17753)
[PASS] testAuthorizeUpgrade() (gas: 3947217)
[PASS] testBurnAmountGreaterThanBalance() (gas: 24858)
[PASS] testBurnZeroAmount() (gas: 30491)
[PASS] testBurnZeroAmountMinusOne() (gas: 6173)
[PASS] testCannotInitializeTwice() (gas: 16053)
[PASS] testDecimals() (gas: 14000)
[PASS] testDefaultAdminRole() (gas: 18567)
[PASS] testDefaultRoles() (gas: 25189)
[PASS] testFuzzBurn(uint256) (runs: 256, μ: 90340, ~: 90450)
[PASS] testFuzzMint(uint96) (runs: 256, μ: 53765, ~: 54120)
[PASS] testFuzzTransfer(uint256) (runs: 256, μ: 52235, ~: 52679)
[PASS] testGrantMinterRole() (gas: 49633)
[PASS] testGrantMinterRoleByNonAdmin() (gas: 37163)
[PASS] testInitialBalance() (gas: 16685)
[PASS] testMintByNonMinter() (gas: 19796)
[PASS] testMintToZeroAddress() (gas: 17593)
[PASS] testMinterRole() (gas: 18363)
[PASS] testName() (gas: 18048)
[PASS] testNonPauserCannotPause() (gas: 19017)
[PASS] testNonUpgraderCannotUpgrade() (gas: 3943016)
[PASS] testPauseUnpause() (gas: 95559)
[PASS] testRevokeMinterRole() (gas: 45080)
[PASS] testRevokeMinterRoleByNonAdmin() (gas: 64850)
[PASS] testSymbol() (gas: 18090)
[PASS] testTransferAmountGreaterThanBalance() (gas: 22872)
[PASS] testTransferToZeroAddress() (gas: 15238)
[PASS] testTransferZeroAmount() (gas: 26918)
[PASS] testTransferZeroAmountMinusOne() (gas: 6209)
Suite result: ok. 29 passed; 0 failed; 0 skipped; finished in 638.99ms (596.65ms CPU time)

Ran 20 tests for test/BurnContract.t.sol:BurnContractTest
[PASS] testAuthorizeUpgrade() (gas: 3033966)
[PASS] testBurnAmountGreaterThanBalance() (gas: 75019)
[PASS] testBurnAmountZero() (gas: 35777)
[PASS] testBurnFailedWhenPaused(uint256) (runs: 256, μ: 123164, ~: 123292)
[PASS] testBurnFailedWhenTokenPaused(uint256) (runs: 256, μ: 139670, ~: 139798)
[PASS] testBurnWhenTotalSupplyIsZero() (gas: 66273)
[PASS] testBurnWithInsufficientAllowance() (gas: 69694)
[PASS] testBurnWithMaxUint256Value() (gas: 74421)
[PASS] testBurnZeroAmountMinusOne() (gas: 24641)
[PASS] testConstructorRevertsWithZeroAddress() (gas: 41051)
[PASS] testFuzzBurn(uint256) (runs: 256, μ: 102541, ~: 102697)
[PASS] testFuzzBurnEvent(uint256) (runs: 256, μ: 100977, ~: 101133)
[PASS] testFuzzMintToRecipient(uint256) (runs: 256, μ: 105036, ~: 105192)
[PASS] testInitializeRevertsWithZeroRecipient() (gas: 7424255)
[PASS] testNonPauserCannotPause() (gas: 18929)
[PASS] testNonUpgraderCannotUpgrade() (gas: 3029720)
[PASS] testPauseUnpause(uint256) (runs: 256, μ: 128471, ~: 128599)
[PASS] testSetRecipient() (gas: 25204)
[PASS] testSetRecipientByNonAdmin() (gas: 19445)
[PASS] testSetRecipientToZeroAddress() (gas: 16859)
Suite result: ok. 20 passed; 0 failed; 0 skipped; finished in 2.88s (2.81s CPU time)

Ran 26 tests for test/DomainNFT.t.sol:DomainNFTTest
[PASS] testAuthorizeUpgrade() (gas: 5816404)
[PASS] testAuthorizeUpgradeByNonAuthorized() (gas: 5814218)
[PASS] testBurnToMint() (gas: 222602)
[PASS] testBurnToMintAlreadyMinted() (gas: 261951)
[PASS] testBurnToMintInvalidSignature() (gas: 123969)
[PASS] testBurnToMintWhenPaused() (gas: 134479)
[PASS] testBurnToMintWhenTotalSupplyBelowTarget() (gas: 110916)
[PASS] testBurnToMintWithoutEnoughBalance() (gas: 133488)
[PASS] testConstructorRevertsWithZeroTokenAddress() (gas: 45679)
[PASS] testInitialize() (gas: 31301)
[PASS] testInitializeRevertsWithZeroBackendSigner() (gas: 10207614)
[PASS] testInitializeRevertsWithZeroRecipient() (gas: 10207567)
[PASS] testPauseByNonAuthorized() (gas: 18821)
[PASS] testPauseUnpause() (gas: 33564)
[PASS] testSetBackendSigner() (gas: 30068)
[PASS] testSetBackendSignerByNonAdmin() (gas: 19289)
[PASS] testSetBackendSignerToZeroAddress() (gas: 16962)
[PASS] testSetRecipient() (gas: 29936)
[PASS] testSetRecipientByNonAdmin() (gas: 19266)

[PASS] testSetRecipientToZeroAddress() (gas: 16790)
[PASS] testSetStakingContractRevertsWithZeroAddress() (gas: 10453945)
[PASS] testSetURI() (gas: 27246)
[PASS] testSetURIByNonAuthorized() (gas: 19855)
[PASS] testSupportsInterface() (gas: 19819)
[PASS] testTransferWhileNotStaked() (gas: 242911)
[PASS] testTransferWhileStaked() (gas: 361675)
Suite result: ok. 26 passed; 0 failed; 0 skipped; finished in 716.65ms (488.53ms CPU time)

Ran 27 tests for test/StakingERC1155Contract.t.sol:StakingERC1155ContractTest
[PASS] testAuthorizeUpgrade() (gas: 3253598)
[PASS] testAuthorizeUpgradeByNonAuthorized() (gas: 3251523)
[PASS] testConstructorRevertsWithZeroStakingTokenAddress() (gas: 3962161)
[PASS] testInitialize() (gas: 18096)
[PASS] testInitializeRevertsWithZeroThawingPeriod() (gas: 13059386)
[PASS] testIsStaking() (gas: 462965)
[PASS] testPauseByNonAuthorized() (gas: 18734)
[PASS] testPauseUnpause() (gas: 33493)
[PASS] testSetDepositAmount() (gas: 29477)
[PASS] testSetDepositAmountToZero() (gas: 16890)
[PASS] testSetThawingPeriod() (gas: 29391)
[PASS] testSetThawingPeriodToZero() (gas: 16868)
[PASS] testSlash() (gas: 336524)
[PASS] testSlashAlreadyThawed() (gas: 329181)
[PASS] testSlashByNonSlasher() (gas: 323490)
[PASS] testSlashNoStake() (gas: 203349)
[PASS] testStake() (gas: 329345)
[PASS] testStakeAlreadyStaked() (gas: 364445)
[PASS] testStakeNotOwner() (gas: 263348)
[PASS] testStakeWhenPaused() (gas: 281872)
[PASS] testStakeWithoutAllowance() (gas: 280469)
[PASS] testSupportsInterface() (gas: 16857)
[PASS] testWithdraw() (gas: 351993)
[PASS] testWithdrawNotOwner() (gas: 331047)
[PASS] testWithdrawStillThawing() (gas: 324418)
[PASS] testWithdrawWhenPaused() (gas: 349544)
[PASS] testWithdrawWithoutStake() (gas: 203626)
Suite result: ok. 27 passed; 0 failed; 0 skipped; finished in 664.92ms (453.09ms CPU time)

Ran 56 tests for testStakingWithDelayedWithdrawalContract.t.sol:StakingWithDelayed WithdrawalContractTest
[PASS] testAuthorizeUpgrade() (gas: 4021145)
[PASS] testConstructorRevertsWithZeroStakingTokenAddress() (gas: 42565)
[PASS] testInitializeRevertsWithZeroStakeAmount() (gas: 8042423)

[PASS] testInitializeRevertsWithZeroThawingPeriod() (gas: 8042402)
[PASS] testNonUpgraderCannotUpgrade() (gas: 4016811)
[PASS] testRequestWithdrawal() (gas: 230257)
[PASS] testRequestWithdrawalAlreadyRequested() (gas: 221231)
[PASS] testRequestWithdrawalNoTokenStake() (gas: 26875)
[PASS] testRequestWithdrawalNotFullyThawed() (gas: 193868)
[PASS] testRequestWithdrawalWrongPermission() (gas: 197546)
[PASS] testRevokeWithdrawal() (gas: 211389)
[PASS] testRevokeWithdrawalNoTokenStake() (gas: 22644)
[PASS] testRevokeWithdrawalNotRequestedYet() (gas: 196222)
[PASS] testRevokeWithdrawalWrongPermission() (gas: 222066)
[PASS] testSetStakeAmount() (gas: 24724)
[PASS] testSetStakeAmountByNonAdmin() (gas: 21718)
[PASS] testSetStakeAmountEvent() (gas: 25254)
[PASS] testSetStakeAmountToZero() (gas: 16769)
[PASS] testSlash() (gas: 191792)
[PASS] testSlashAfterStakeWithPermit() (gas: 262070)
[PASS] testSlashEvent() (gas: 182282)
[PASS] testSlashFailedWhenPaused() (gas: 217743)
[PASS] testSlashFailedWhenTokenPaused() (gas: 224016)
[PASS] testSlashFailsWhenConsideredWithdrawn() (gas: 226474)
[PASS] testSlashFromNonAdmin() (gas: 200751)
[PASS] testSlashingZeroToke() (gas: 23506)
[PASS] testStake() (gas: 204540)
[PASS] testStakeBelowMinStakeAmount() (gas: 180417)
[PASS] testStakeEvent() (gas: 193378)
[PASS] testStakeFailedWhenPaused() (gas: 213334)
[PASS] testStakeFailedWhenTokenPaused() (gas: 343327)
[PASS] testStakeTwice() (gas: 213464)
[PASS] testStakeWithNotEnoughFunds() (gas: 214238)
[PASS] testStakeWithPermit() (gas: 268193)
[PASS] testStakeWithPermitBelowStakeAmount() (gas: 81032)
[PASS] testStakeWithPermitWithWrongContract() (gas: 78565)
[PASS] testStakeWithPermitWithoutFund() (gas: 236088)
[PASS] testStakeWithoutEnoughApproval() (gas: 180691)
[PASS] testUpdateThawingPeriod() (gas: 25266)
[PASS] testUpdateThawingPeriodByNonAdmin() (gas: 22077)
[PASS] testUpdateThawingPeriodEvent() (gas: 27769)
[PASS] testUpdateThawingPeriodToZero() (gas: 16956)
[PASS] testUpdateWithdrawalPeriod() (gas: 25109)
[PASS] testUpdateWithdrawalPeriodByNonAdmin() (gas: 21964)
[PASS] testUpdateWithdrawalPeriodEvent() (gas: 27656)
[PASS] testUpdateWithdrawalPeriodToZero() (gas: 16823)

[PASS] testWithdraw() (gas: 204468)
[PASS] testWithdrawAsOwnerAfterStakeWithPermit() (gas: 284422)
[PASS] testWithdrawAsStakerAfterStakeWithPermit() (gas: 286662)
[PASS] testWithdrawNotRequestedYet() (gas: 196113)
[PASS] testWithdrawWithdrawPeriodNotPassed() (gas: 223947)
[PASS] testWithdrawWrongWallet() (gas: 224565)
[PASS] testWithdrawnEvent() (gas: 198518)
[PASS] testWithdrawnFailedWhenPaused() (gas: 228780)
[PASS] testWithdrawnFailedWhenTokenPaused() (gas: 233872)
[PASS] testWithdrawnFails() (gas: 193504)
Suite result: ok. 56 passed; 0 failed; 0 skipped; finished in 399.48ms (270.89ms CPU time)

Ran 40 tests for test/StakingContract.t.sol:StakingContractTest
[PASS] testAuthorizeUpgrade() (gas: 3452623)
[PASS] testConstructorRevertsWithZeroTokenAddress() (gas: 41664)
[PASS] testInitializeRevertsWithZeroStakeAmount() (gas: 7842741)
[PASS] testInitializeRevertsWithZeroThawingPeriod() (gas: 7842698)
[PASS] testNonUpgraderCannotUpgrade() (gas: 3448378)
[PASS] testSetStakeAmount() (gas: 24679)
[PASS] testSetStakeAmountByNonAdmin() (gas: 21696)
[PASS] testSetStakeAmountEvent() (gas: 25210)
[PASS] testSetStakeAmountToZero() (gas: 16791)
[PASS] testSlash() (gas: 169370)
[PASS] testSlashAfterStakeWithPermit() (gas: 239683)
[PASS] testSlashEvent() (gas: 160932)
[PASS] testSlashFailedWhenPaused() (gas: 195285)
[PASS] testSlashFailedWhenTokenPaused() (gas: 201911)
[PASS] testSlashFromNonAdmin() (gas: 174334)
[PASS] testSlashingZeroToke() (gas: 21316)
[PASS] testStake() (gas: 173839)
[PASS] testStakeBelowMinStakeAmount() (gas: 153956)
[PASS] testStakeEvent() (gas: 166850)
[PASS] testStakeFailedWhenPaused() (gas: 186066)
[PASS] testStakeFailedWhenTokenPaused() (gas: 289599)
[PASS] testStakeTwice() (gas: 187045)
[PASS] testStakeWithNotEnoughFunds() (gas: 183015)
[PASS] testStakeWithPermit() (gas: 237563)
[PASS] testStakeWithPermitBelowStakeAmount() (gas: 81032)
[PASS] testStakeWithPermitWithWrongContract() (gas: 78587)
[PASS] testStakeWithPermitWithoutFund() (gas: 209605)
[PASS] testStakeWithoutEnoughApproval() (gas: 154207)
[PASS] testUpdateThawingPeriod() (gas: 25266)
[PASS] testUpdateThawingPeriodByNonAdmin() (gas: 22033)
[PASS] testUpdateThawingPeriodEvent() (gas: 27769)

[PASS] testUpdateThawingPeriodToZero() (gas: 16911)
[PASS] testWithdraw() (gas: 161449)
[PASS] testWithdrawAsOwnerAfterStakeWithPermit() (gas: 240408)
[PASS] testWithdrawAsStakerAfterStakeWithPermit() (gas: 243643)
[PASS] testWithdrawWrongWallet() (gas: 170741)
[PASS] testWithdrawnEvent() (gas: 155534)
[PASS] testWithdrawnFailedWhenPaused() (gas: 185760)
[PASS] testWithdrawnFailedWhenTokenPaused() (gas: 190787)
[PASS] testWithdrawnFails() (gas: 167110)
Suite result: ok. 40 passed; 0 failed; 0 skipped; finished in 350.39ms (250.32ms CPU time)

Ran 24 tests for test/RewardLiquidityPoolContract.t.sol:RewardPoolContractTest
[PASS] testClaim() (gas: 127244)
[PASS] testClaimNoReward() (gas: 20973)
[PASS] testClaimWhenPaused() (gas: 156499)
[PASS] testConstructorRevertsWithZeroTokenAddress() (gas: 41330)
[PASS] testFundContract() (gas: 81018)
[PASS] testIncreaseDynamicAmount() (gas: 158978)
[PASS] testIncreaseDynamicAmount100000Recipients() (gas: 944201810)
[PASS] testIncreaseDynamicAmount10000Recipients() (gas: 52342680)
[PASS] testIncreaseDynamicAmount1000Recipients() (gas: 4922337)
[PASS] testIncreaseDynamicAmount100Recipients() (gas: 597921)
[PASS] testIncreaseDynamicAmount10Recipients() (gas: 169706)
[PASS] testIncreaseDynamicAmountNoRecipients() (gas: 63743)
[PASS] testIncreaseDynamicAmountNotEnoughFund() (gas: 122929)
[PASS] testIncreaseDynamicAmountRecipientsAmountsMismatch() (gas: 69406)
[PASS] testIncreaseDynamicAmountTo0x() (gas: 68346)
[PASS] testIncreaseFixedAmount() (gas: 154712)
[PASS] testIncreaseFixedAmount100000Recipients() (gas: 441721263)
[PASS] testIncreaseFixedAmount10000Recipients() (gas: 33734432)
[PASS] testIncreaseFixedAmount1000Recipients() (gas: 3377151)
[PASS] testIncreaseFixedAmount100Recipients() (gas: 445816)
[PASS] testIncreaseFixedAmount10Recipients() (gas: 153684)
[PASS] testIncreaseFixedAmountNoRecipients() (gas: 63431)
[PASS] testIncreaseFixedAmountNotEnoughFund() (gas: 75033)
[PASS] testIncreaseFixedAmountTo0x() (gas: 73008)
Suite result: ok. 24 passed; 0 failed; 0 skipped; finished in 11.47s (15.76s CPU time)

Ran 7 test suites in 11.48s (17.12s CPU time): 222 tests passed, 0 failed, 0 skipped (222 total tests)

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

FILE	% STMTS	% BRANCH	% FUNCS	% LINES
AukiToken.sol	100	100	100	100
BurnContract.sol	100	100	100	100
DomainNFT.sol	100	100	100	100
RewardLiquidityPoolContract.sol	100	100	100	100
StakingContract.sol	100	100	100	100
StakingERC1155Contract.sol	100	100	100	100
StakingWithDelayedWithdrawal Contract.sol	100	100	100	100
UUPSProxy.sol	100	100	100	100
All Files	100	100	100	100

We are grateful for the opportunity to work with the Auki Labs team.

The statements made in this document should not be interpreted as an investment or legal advice, nor should its authors be held accountable for the decisions made based on them.

Zokyo Security recommends the Auki Labs team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo