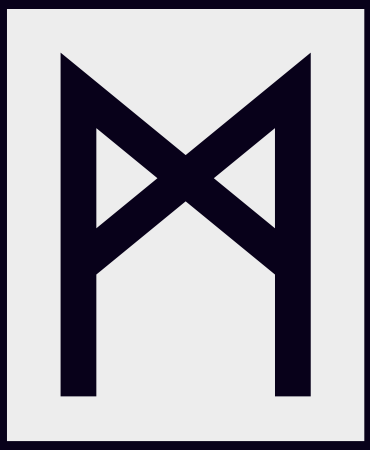


AUKI

SMART CONTRACTS REVIEW

zokyo

October 2nd 2023 | v. 1.0

Security Audit Score

PASS

Zokyo Security has concluded that these smart contracts passed security review.

TECHNICAL SUMMARY

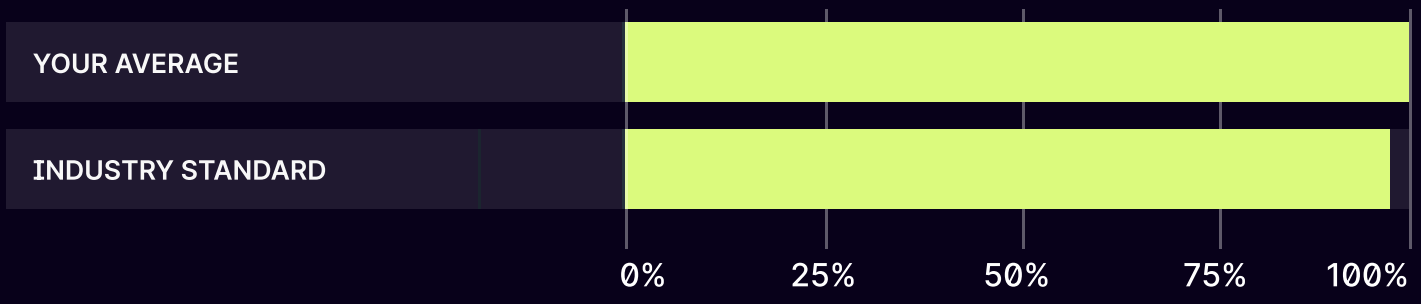
This document outlines the overall security of the Auki Labs smart contracts evaluated by the Zokyo Security team.

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

Contract Status

There were 0 critical issues found during the audit. (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 contracts 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 contracts.

Table of Contents

AUDITING STRATEGY AND TECHNIQUES APPLIED

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

Last commit - da7c04790790fb997fd9e958cf69a60eb277b73b

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

- AukiToken.sol
- FreeStorageWallet.sol
- StakingContract.sol
- BurnContract.sol
- FreeStorageWalletWhitelist.sol
- UUPSProxy.sol
- RewardLiquidityPoolContract.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 contracts. To do so, the code was reviewed line by line by our smart contract developers, who documented even minor issues as they were discovered. Part of this work includes writing a test suite using the Hardhat 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.
02	Cross-comparison with other, similar smart contracts by industry leaders.
03	Testing contracts logic against common and uncommon attack vectors.
04	Thorough manual review of the codebase line by line.

Executive Summary

The Zokyo team conducted an in-depth review of Auki Labs' codebases. During the review, we identified one medium-level issue and two informational issues. The Auki Labs team promptly addressed all of these findings. You can find detailed descriptions of these resolutions in the "Complete Analysis" section.

Zokyo Audit Scoring Auki Labs:

1. **Severity of Issues:** (Critical, High, Medium, Low and Informational)
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.

Hypothetical 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
- 1 Medium issue: -10 points
- 2 Informational issues: -2 points

Thus, 100 - 10 - 2 = 88.

However, with 100% test coverage (above the industry standard of 95%) +10 for the excellent test coverage.

So, 88 + 10 = 98.

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	Possible underflow in supply calculations	Medium	Resolved
2	Require statements without a revert message	Informational	Resolved
3	Upgradeable contracts don’t have storage gaps	Informational	Resolved

MEDIUM-1 | RESOLVED

Possible underflow in supply calculations

In contract, BurnContract.sol, in function burn, currentSupplySubTargetSupply is calculated by subtracting TARGET_TOTAL_SUPPLY from currentTotalSupply(total supply of AukiToken). If currentTotalSupply is smaller than TARGET_TOTAL_SUPPLY the subtraction will underflow. Even if it is intended behavior for the function to revert in this case, the revert should be triggered by a required statement with a proper error message.

Recommendation:
Add a sanity check to ensure that currentTotalSupply is greater than TARGET_TOTAL_SUPPLY and revert with a proper message otherwise.

INFORMATIONAL-1 | RESOLVED

Require statements without a revert message

In contract, FreeStorageWallet at lines #34 & #35 there are two require statements that don’t have a revert message.

Recommendation:
Add a revert message to all the require statements

Upgradeable contracts don’t have storage gaps

All the contracts also have the possibility to be upgradeable as they are implementing the UUPS pattern, however, they are not implementing storage gaps to ensure there are no storage conflicts between possible upgrade

Recommendation:
Implement storage gaps in all the upgradeable contracts.

	AukiTOken.sol FreeStorageWallet.sol StakingContract.sol BurnContract.sol FreeStorageWalletWhitelist.sol UUPSProxy.sol RewardLiquidityPoolContract.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 contracts code, our team was responsible for writing integration tests using the Hardhat testing framework.

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

AukiToken

pause
✓ pause (64ms)
pause
✓ pause (60ms)
✓ pause
✓ pause (105ms)
mint
✓ mint (43ms)
✓ mint (100ms)
mint
✓ mint (38ms)
_disableInitializers
✓ _disableInitializers (238ms)

BurnContract

true
✓ deploy true (288ms)
initialize
✓ initialize (45ms)
✓ initialize (38ms)
✓ initialize (102ms)
pause
✓ pause (94ms)
✓ pause (143ms)
✓ pause (115ms)
burn
✓ burn (109ms)
✓ burn (81ms)
✓ burn (117ms)

setRecipient
✓ setRecipient (60ms)
✓ setRecipient (229ms)
✓ setRecipient (54ms)

FreeStorageWallet

deployment
✓ deployment (39ms)
✓ deployment (254ms)
withdraw
✓ withdraw
✓ withdraw
✓ withdraw
✓ withdraw
transferOwnership
✓ transferOwnership
renounceOwnership
✓ renounceOwnership

FreeStorageWalletWhitelist

addSpender
✓ addSpender
✓ addSpender (80ms)
✓ addSpender (45ms)
✓ removeSpender (62ms)
✓ removeSpender
✓ removeSpender (42ms)
addSpender
✓ addSpender
requestWithdrawal
✓ requestWithdrawal (79ms)
✓ requestWithdrawal (74ms)
✓ requestWithdrawal (65ms)
✓ requestWithdrawal (165ms)
✓ requestWithdrawal (48ms)
✓ requestWithdrawal (98ms)
withdraw
✓ withdraw (59ms)
✓ withdraw
✓ withdraw (74ms)
✓ withdraw (52ms)
✓ withdraw (51ms)
✓ withdraw (192ms)

RewardLiquidityPoolContract

fundContract

✓ fundContract

✓ fundContract

claim

✓ claim

✓ claim

✓ claim

increaseFixedAmount

✓ increaseFixedAmount (80ms)

✓ increaseFixedAmount (79ms)

✓ increaseFixedAmount (66ms)

✓ increaseFixedAmount

✓ increaseFixedAmount

increaseFixedAmount

✓ increaseFixedAmount (72ms)

✓ increaseFixedAmount (225ms)

✓ increaseFixedAmount (62ms)

✓ increaseFixedAmount (52ms)

pendingReward

✓ pendingReward

StakingContract

should fail deployment

✓ should fail deployment

initialize

✓ initialize (264ms)

✓ initialize (343ms)

✓ initialize (294ms)

setThawingPeriod

✓ setThawingPeriod (48ms)

✓ setThawingPeriod (60ms)

✓ setThawingPeriod (51ms)

setStakeAmount

✓ setStakeAmount (49ms)

✓ setStakeAmount (51ms)

✓ setStakeAmount (62ms)

pause

✓ pause (72ms)

✓ pause (163ms)

stake

✓ stake

✓ stake

✓ stake (69ms)
✓ stake (53ms)
✓ stake
✓ stake
withdraw
✓ withdraw (39ms)
✓ withdraw
✓ withdraw
✓ withdraw
✓ withdraw (74ms)
✓ withdraw (45ms)
slash
✓ slash (77ms)
✓ slash (144ms)
✓ slash (59ms)
✓ slash (108ms)

91 passing (33s)

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