Formal Verification Report Aquarius AMM

Cantina Aquarius Contest - Formal Verification Section

Auditor	Method	Date
Asaf Dov “ArlidenTheBard”	Certora Soroban Prover Sunbeam (7.29.2)	May – June 2025

Table of Contents

1. Executive Summary

This report details the findings of a formal verification engagement on the Aquarius smart contract suite, conducted between May and June 2025, in the context of the formal verification section of the respective Cantina contest. The primary goal of this audit was to create a comprehensive formal specification to rigorously test the logic and security of the target smart contracts.

The audit identified a high impact vulnerability that allows the Admin role to be assigned to the contract's own address, which could render the contract inoperable. Furthermore, the audit identifies three tool related findings which include logical flaws in role management that prevent the Admin role from being transferred and compromise interactions for roles with multiple users.

This report provides detailed descriptions of these vulnerabilities, their potential impact, and recommendations for remediation. It also includes additional architectural suggestions to enhance the system's robustness and a list of key properties that were successfully verified.

2. Audit Scope & Methodology

2.1. Scope

The following smart contracts and libraries were in scope for this formal verification audit:

- **fees_collector/contract.rs**: The main smart contract for fee collector management.
- **access_control/**: A library for managing roles and permissions.
- **Upgrade/**: A library for handling contract upgrades.

2.2. Methodology

This audit was conducted using the Certora Soroban Prover “Sunbeam” (7.29.2) to formally verify the contract's behavior against a comprehensive set of rules.

Note: The approach was to write a complete specification file, exceeding the contest's minimum requirements, to ensure robust coverage.

To facilitate a thorough analysis, several custom utility functions were developed to supplement missing functionality in the Soroban-sdk at the time of the audit. These helper functions, located in **fees_collector/src/certora_specs/utils_ext.rs**, include:

Function Name	Description
nondet_role	Returns a non-deterministic role
nondet_wasm	Returns a non-deterministic WASM hash for contract upgrade rules
nondet_func	A parametric function implementation that calls a function with non-deterministic inputs, implemented separately for the access control library and the fee collector contract
role_to_string	A helpful function to log the role for debugging

The specification rules were separated by components for clarity, with **fees_collector rules** and **access_control rules** located in their respective files.

2.3. General Assumptions and Simplifications

To enable the prover to access necessary contract states, minor visibility changes were made to the source code, in accordance with this discord help-desk post. The following source code changes were made:

1) access_control/src/lib.rs – changed visibility of storage mod to pub
2) access_control/src/roles.rs – changed visibility of the following to public
    a. fn *has_many_users()*
    b. fn *is_transfer_delayed()*
3) access_control/src/storage.rs – changed visibility of the following to pub
    a. *enum DataKey*
    b. *trait StorageTrait*
4) upgrade/src/lib.rs – changed the visibility of storage mod to pub
5) upgrade/src/storage – changed the visibility of *DataKey*

2.4. Spec File Layout

The file structure of the Aquarius repository relevant to this audit includes:

Specification files:

- **Fees Collector** smart contract rules are in the original file located: fees_collector/src/certora_specs/fee_collector_rules.rs.
- **Access Control** library rules are located separately in the following location: fees_collector/src/certora_specs/access_control_rules.rs

Configuration files:

- fees_collector_verified.conf – Contains all the verified Fees Collector rules, without the violated ones.
- access_control_verified.conf – Contains all the verified Access Control rules, without the violated ones.
- fees_collector_vec_comparison_bug_violated.conf – Contains violated and verified rules pertaining to the Vec Comparison bug
- fees_collector_from_symbol_bug_violated.conf – Contains violated and verified rules pertaining to the From_Symbol bug
- fees_collector_contract_address_bug_violated.conf – Contains violated and verified rules pertaining to the Contract Address bug
- access_control_vec_contains_bug_violated.conf – Contains violated and verified rules pertaining to the Vec.contains bug

Utilities:

- fees_collector/src/certora_specs/utils_ext.rs

3. Summary of Findings

3.1. System Vulnerabilities

Real vulnerabilities found and submitted through Cantina.

Impact	Likelihood	Title	Finding
High	Low	Contract Can Be Permanently Locked	The Admin role can be transferred to the contract's own address, which lacks built-in recovery mechanisms.

3.2. Tool Related Findings

These are bugs that have been found by violation of certain properties. However, when running tests on the deployed contract in the native rust environment, the violated functionality was found to be working as intended.

Impact	Title	Finding
High	Admin Role Transfer Failure	A bug in Role::from_symbol() prevents the Admin role from being transferred due to a symbol comparison flaw.
High	Flawed Multi-User Role Management	The Vec.contains function is not implemented, compromising functions that manage roles with multiple assigned users.
Low	Incorrect Vector Comparison	Comparing two newly created vector instances returns false even if their contents are identical, leading to potential logic errors.

4. Detailed Findings

4.1. System Vulnerabilities

4.1.1. Contract Address Bug

- **Severity**: Medium
  (Medium, according to Catina's severity map given high impact with low likelihood)
- **Description**: It is possible to assign or transfer the Admin role to the smart contract's own address. The Soroban SDK documentation suggests that is_auth(contract address) should revert, but this was not the observed behavior.
- **Impact**: This action effectively locks all administrative functions. Without a designated emergency_admin or other recovery mechanism, the contract becomes permanently unmanageable and un-upgradable, rendering it useless.
- **Evidence**: The property contract_cant_have_role failed during verification, proving this scenario is possible.

Recommendation: Implement a check in all role-transfer and initialization functions to explicitly forbid assigning a role to the contract's own address (env.current_contract_address()) as shown in Github Issue #2.

4.2. Tool Related Findings

Below are the details of the tool related findings. Severity has been classified as if the findings were system bugs.

4.2.1. Vec.contains Bug

- **Severity**: High – for breaking core functionality
- **Description**: The .contains() method for vector types is not fully implemented in the verification environment.
- **Impact**: This bug compromises all interactions with roles that support multiple users. Core functions designed to check permissions for these roles will fail to operate as intended. The affected functions include:
  - access_control::access::address_has_role
  - access_control::access::assert_address_has_role
  - access_control::utils::require_pause_or_emergency_pause_admin_or_owner
- **Evidence**: Unit tests for the affected functions, such as address_has_role_integrity and assert_address_has_role_integrity, consistently fail.

Recommendation: This appears to be a limitation of the tooling environment. The issue has been brought to the attention of the tool developers in this help-desk discord post. As previously suggested, this is not a real bug. See Github issue #1 for working test.

4.2.2 Role::from_symbol() Bug

- **Severity**: High – for breaking core functionality
- **Description**: The function access_control::role::Role::from_symbol() reverts when processing the symbol for Role::Admin. The issue stems from an incorrect comparison of symbols that are nine characters or less.
- **Impact**: This bug prevents the Admin from executing critical ownership transfer functions, including the following:
    - Fees_collector::commit_transfer_ownership
    - Fees_collector::apply_transfer_ownership
    - Fees_collector::revert_transfer_ownership
    - Fees_collector::get_future_address

  While other roles can be transferred, the primary administrative role is immobilized.
- **Evidence**:
    - The rule admin_can_transfer fails, demonstrating the inability to transfer the Admin role.
    - The rule role_from_symbol_reverts_for_admin demonstrates the issue being specifically with the from_symbol function.
    - A temporary fix of changing the "Admin" symbol to "ContractAdmin" allowed the rule to pass, isolating the issue to the symbol itself.
    - The corresponding rule admin_can_transfer_access_control passes, confirming the issue is specific to the fees_collector implementation context

Recommendation: The underlying symbol comparison logic in Role::from_symbol() should be corrected. As a temporary workaround, rename the Admin role to a string longer than nine characters, such as "ContractAdmin", as seen in Github issue #4. Note that the issue shows this isn't a real application bug, but a tool environment bug.

4.2.3 Vec Comparison Bug

- **Severity**: Low/Informational – Not necessary for the current implementation, as it isn't used.
- **Description**: When access_control::management::get_role_addresses() returns an empty vector, it does so by creating a new vector instance via unwrap_or(Vec::new(&e)). When comparing two such vectors, the `==` operator incorrectly returns false because it compares the vector instances, not their contents.
- **Impact**: Rules and logic that depend on comparing address lists from get_role_addresses may behave incorrectly, especially in edge cases involving empty roles. This could lead to failed assertions or incorrect state transitions.
- **Evidence**: This behavior was confirmed by the rules
    - compare_two_poissibly_empty_vectors
    - compare_two_vectors_at_least_one_isnt_empty

Recommendation: When comparing vectors of addresses returned by get_role_addresses, check their length as well, or implement element-wise comparison. The solution proposed in Github issue #3 is to instantiate any role that has many users with an empty vector int the contracts constructor. Note that the issue shows this isn't a real application bug, but a tool environment bug.

5. Additional Recommendations

- **Two-Step Role Transfer:** For critical roles like Admin, consider implementing a two-step transfer process where the proposed new address must provide a signature to accept the role before the transfer is finalized. This would prevent accidental transfers to incorrect or inaccessible addresses.

- **Configurable Delay Constants:** The time delays for ownership transfers and contract upgrades are currently constants. This lack of flexibility can be problematic. For instance, if an owner mistakenly initiates a role transfer to a wrong address with a one-year delay, the contract administration is frozen for that period, assuming emergency_admin was not previously assigned. It is recommended to make these delay periods configurable by an administrative role.

6. Verified Properties

Beyond the vulnerabilities found, the formal verification process successfully validated numerous properties of the system, confirming that the contract behaves as expected in many scenarios. Key contract functionalities validated include:

- **Emergency Mode**
- **Initialization**
- **Role Transfer Logic**
    - Roles with single or multiple users
    - Assignment of future roles
    - Deadline integrity
    - Revert scenerios
- **Contract Upgrade Logic:**
    - Future wasm hash
    - Deadline integrity
    - Revert scenarios

A complete list of all properties and rules checked can be found in Appendix A.

This concludes the formal verification report for the Aquarius application.

Asaf
Security Researcher

7. Appendix A: Full list of properties with links to the Prover reports

Property	Function Name/Comment
Fees Collector Rules	
Emergency mode changed => EmergencyAdmin != None	emergency_mode_changed_emergency_admin_is_some
Emergency mode changed => Emergency admin called set emergency mode	emergency_mode_state_transition
Admin not None => Can't call init_admin()	no_init_if_admin_exists
Roles and Role Transfer Logic	
One address per role unless role.has_many_users	one_address_per_role
Role changed => apply transfer was called	role_only_changes_if_apply_transfer
Revert called => apply does nothing	cant_apply_transfer_if_revert_called
Role transfered after apply => (deadline <= blocktimestamp)	role_cant_transfer_within_deadline
Admin can transfer his role	admin_can_transfer Note: rule fails as is unreachable for Admin. See from_symbol() bug
Emergency Admin can transfer his role	emergency_admin_can_transfer Note: This rule was only written to prove that the role transferring issue is with Admin only.
Cant transfer role to None	cant_transfer_role_to_none
Role has deadline => role.is_transfer_delayed	role_has_deadline_is_transfer_delayed
Role.is_transfer_delayed => Role has a deadline	role_is_transfer_delayed_has_deadline
Transferring a role doesn't affect the other roles	To save time I separated cases of role and other role has_many_users or not: • one_role_at_a_time_both_not_has_many_ • one_role_at_a_time_transfering_role_has_many_ • one_role_at_a_time_other_role_has_many_users – Fails, fue • one_role_at_a_time_both_has_many_users
Transfer deadline changed from zero => deadline > current time	deadline_state_transition_transfer Written under the same rule
Transfer deadline changed from nonzero => deadline = 0	
Transfer Deadline changed to nonzero value => commit was called	deadline_changed_due_to_commit

Transfer Deadline changed to zero value => apply or revert was called	deadline_changed_due_to_revert_or_apply
Transfer Deadline > 0 => commit transfer reverts	cant_commit_before_deadline_transfer
Transfer deadline can only change zero or now()<deadline	deadline_valid_states_fees_collector
Only Admin can call transfer or upgrade functions (commit, apply, revert)	only_admin_transfers_roles_or_upgrades
Role transferred => role Must be only Admin or Emergency Admin.	only_admin_or_emergency_admin_be_transfered Note: Fails for Admin due to the from_symbol() bug. See this run See admin_can_transfer rule
Contract address can't have a role	contract_cant_have_role Fails. See contract address bug
if future address changed => someone called commit	future_address_state_transition
Future address changed => transfer deadline changed	future_address_changed_deadline_changed_fees_collector
Future address cant become None	future_address_cant_become_none_fees_collector
Contract Upgrade Logic	
Emergency mode => Admin can apply upgrade disregarding deadline	no_upgrade_delay_if_emergency_mode
Future wasm changed => Admin called commit, and future wasm is not none	future_wasm_state_transition
future wasm changed => deadline changed	future_wasm_changed_deadline_changed
future wasm cant become None	future_wasm_cant_be_none
Upgrade deadline changed to nonzero value => commit upgrade was called	deadline_changed_due_to_commit_upgrade
Upgrade deadline changed to zero value => applyUpgrade or revertUpgrade was called	deadline_changed_due_to_revert_or_apply_upgrade
Upgrade deadline changed from 0 => current time < deadline	deadline_state_transition_upgrade
Upgrade deadline changed nonzero => deadline = 0	
Upgrade deadline != 0 => commit reverts	cant_commit_if_deadline_nonzero_upgrade

Unit Tests	
commit_upgrade_integrity	commit_upgrade_integrity
apply_upgrade_integrity	apply_upgrade_integrity
revert_upgrade_integrity	revert_upgrade_integrity
set_emergency_mode_integrity	set_emergency_mode_integrity
commit_transfer_ownership_integrity	commit_transfer_ownership_integrity
apply_transfer_ownership_integrity	apply_transfer_ownership_integrity
revert_transfer_ownership_integrity	revert_transfer_ownership_integrity
get_future_address_integrity_admin	get_future_address_integrity_admin Written to prove the function reverts for Admin. See from_symbol() bug
get_future_address_integrity	get_future_address_integrity_fees_collector
get_emergency_mode_integrity	get_emergency_mode_integrity
Role.as_symbol reverts for Admin	role_from_symbol_reverts_for_admin

Continue next page

Access Control Rules	
Emergency mode changed => set_emergency_mode called	emergency_mode_state_transition_access_control
Role.has_many_user => get_role_safe reverts	role_has_many_users_get_role_safe_reverts
Role.has_many_user => get_role reverts	role_has_many_users_get_role_reverts
Role.has_many_user => set_role_address reverts	role_has_many_users_set_role_address_reverts
!role.has_many_user => set_role_addresses reverts	role_not_has_many_users_set_role_addresses_reverts
!role.has_many_user => get_role_addresses reverts	role_not_has_many_users_get_role_addresses_reverts
Future address changed => deadline changed	future_address_changed_deadline_changed_access_control
Future address changed => commit was called	future_address_state_transition_access_control
Future address cant become none	future_address_cant_become_none_access_control
Deadline changed to nonzero => commit or put_transfer_deadline called	deadline_changed_to_nonzero_commit_or_put_transfer_deadline
Deadline changed to zero => revert, apply or put_transfer_deadline	deadline_changed_to_zero_revert_apply_or_put_transfer_deadline
Deadline changed => deadline > now() or deadline = 0	deadline_valid_states_access_control
Deadline != 0 => commit reverts	cant_commit_if_deadline_nonzero
Now() < deadline and role address is not none => apply reverts	cant_apply_before_deadline
Role address changed => new address == future address	role_changed_future_address_is_new_address
Role cant change to None	role_cant_change_to_none
Role changed => role is either Admin or EmergencyAdmin unless using set_role_address/es	role_changed_is_admin_or_emergency_admin Fails when role is Admin due to the from_symbol bug. See this run
Role changed and role.has_many_users => set_role_addresses was called.	role_has_many_users_changes_due_to_set_role_addresses
role.transfer delay => !role.has_many_users	role_has_transfer_delay_has_one_user

Role changed => apply was called or set_role_address/es	role_changed_due_to_apply_or_set_role
Transfering a role doesnt affect the other roles	To save time I separated cases of role and other role has_many_users or not: • one_role_at_a_time_both_not_has_many_access_control • one_role_at_a_time_transfering_role_has_many_users • one_role_at_a_time_other_role_has_many_users_access_control • one_role_at_a_time_both_has_many_users_access_control
Admin can transfer his role	admin_can_transfer_access_control Rule passes. Proving the issue is in fees_collector. See form_symbol bug for more.
Emergency Admin can transfer his role	emergency_admin_can_transfer_access_control Might be redundant, but written due to the from_symbol bug.
Revert called => apply transfer reverts	cant_apply_transfer_if_revert_called_access_control

Continue next page

Unit Tests	
address_has_role integrity	address_has_role_integrity Fails due to the Vec.contains bug
get_role_safe integrity	get_role_safe_integrity
get_role integrity	get_role_integrity
set_role_address integrity	set_role_address_integrity
get_role_addresses_integrity	get_role_addresses_integrity
set_role_addresses_integrity	set_role_addresses_integrity
get_emergency_mode_integrity_access_control	get_emergency_mode_integrity_access_control
require_rewards_admin_or_owner_integrity	require_rewards_admin_or_owner_integrity
require_operations_admin_or_owner_integrity	require_operations_admin_or_owner_integrity
require_pause_or_emergency_pause_admin_or_owner_integrity	require_pause_or_emergency_pause_admin_or_owner_integrity Fails due to the Vec.contains bug
require_pause_admin_or_owner_integrity	require_pause_admin_or_owner_integrity
assert_address_has_role_integrity	assert_address_has_role_integrity Fails due to the Vec.contains bug
commit_transfer_ownership_integrity_access_control	commit_transfer_ownership_integrity_access_control
apply_transfer_ownership_integrity_access_control	apply_transfer_ownership_integrity_access_control
revert_transfer_ownership_integrity_access_control	revert_transfer_ownership_integrity_access_control
get_future_address_integrity	get_future_address_integrity
put_transfer_ownership_deadline_integrity	put_transfer_ownership_deadline_integrity
get_transfer_ownership_deadline_integrity	get_transfer_ownership_deadline_integrity