Aquarius Audit

March 2024

By CoinFabrik

v202311

Executive Summary

CoinFabrik was asked to audit the contracts for the Aquarius project.

During this audit we found two medium issues and several minor issues. Also, several enhancements were proposed. All the issues were resolved except for one minor issue that was acknowledged. Most of the proposed enhancements were implemented.

Scope

The audited files are from the git repository located at https://github.com/AquaToken/soroban-amm.git.

The audit was conducted (in order) on the following commits:

1. `8c481ea41c11f23db546d81071bb32d2763161d7` (`v1.1.1` tag)
2. `85aab00b392170433eb82235e37a9160de4376b5` (`audit-v3` tag)
3. `afd7e3fb40c92fbeafb322e9bba7a6f3a29eba71` (`audit-v4` tag)
4. `aaa94a13cd3ff072788dc82a036d836787b54567` (`audit-v5` tag)
5. `02c79a24b0314abd5c52c454213c7bd3f152bad6` (`audit-v6` tag)
6. `de33577d289c008862a5b8b0bc561e6802c21ecc` (`audit-v9` tag)
7. `f1f1f35d702b8283243f9bace37c1f84cd753950` (`audit-v10` tag)

The status of the issues and enhancements reported corresponds to the last commit in the list.

The scope for this audit includes and is limited to the following files:

- `access_control` directory: Library that supports having administrators in a contract.
    - `src/access.rs`: `AccessControl` object implementation.
    - `src/lib.rs`: Exports functionality implemented in the library.
- `liquidity_pool` directory: `soroban-liquidity-pool-contract` implementation. It is an exchange liquidity pool based on constant product formula (x*y=k).
    - `src/constants.rs`: `FEE_MULTIPLIER` constant definition.
    - `src/contract.rs`: Contract implementation.
    - `src/lib.rs`: Exports contract and client.
    - `src/plane_interface.rs`: Plane contract trait definition.
    - `src/plane.rs`: Utility functions to interact with the pool plane.
    - `src/pool_interface.rs`: Traits definitions for the contract.
    - `src/pool.rs`: Contains a function to calculate amounts to deposit.
    - `src/rewards.rs`: Contains the `get_rewards_manager` function.
    - `src/storage.rs`: Utility functions to manage the contract storage.
    - `src/token.rs`: Utility functions to interact with token contracts.

- `liquidity_pool_plane` directory: `soroban-liquidity-pool-plane-contract` implementation. It is a lightweight contract which is purposed to store actual information about every pool. It allows the system to emulate pool calculations.
  - `src/contract.rs`: Contract implementation.
  - `src/interface.rs`: Contract trait definition.
  - `src/lib.rs`: Exports contract and client.
  - `src/storage.rs`: Functions to persist plane data.
- `liquidity_pool_router` directory: `soroban-liquidity-pool-router-contract` implementation. It works as an entry point and catalog of liquidity pools. It is capable of deploying new pools if necessary.
  - `src/constants.rs`: Constant definitions.
  - `src/contract.rs`: Contract implementation.
  - `src/events.rs`: Utility functions to publish events.
  - `src/lib.rs`: Exports contract and client.
  - `src/pool_contract.rs`: standard liquidity pool contract client.
  - `src/pool_interface.rs`: Traits definitions for pools, plane and swap router.
  - `src/pool_utils.rs`: utility functions to interact with the pools.
  - `src/rewards.rs`: provide `get_rewards_manager` function.
  - `src/router_interface.rs`: Traits definitions for the contract.
  - `src/storage.rs`: Utility functions to manage the contract storage.
  - `src/swap_router.rs`: `soroban-liquidity-pool-swap-router-contract` client definition.
- `liquidity_pool_stableswap` directory: `soroban-liquidity-pool-stableswap-contract` implementation. It is a curve.fi-like exchange contract.
  - `src/contract.rs`: Contract implementation.
  - `src/lib.rs`: Exports contract definitions.
  - `src/plane_interface.rs`: Plane contract trait.
  - `src/plane.rs`: Utilities to interact with the `liquidity_pool_plane` contract.
  - `src/pool_2_constants.rs`: Constants used when the project is compiled with support for 2 coins.
  - `src/pool_3_constants.rs`: Constants used when the project is compiled with support for 3 coins.
  - `src/pool_4_constants.rs`: Constants used when the project is compiled with support for 4 coins..
  - `src/pool_constants.rs`: Project constants. Some constants vary depending on how many coins are chosen in the compilation.
  - `src/pool_interface.rs`: Trait definitions for the contract.
  - `src/rewards.rs`: provide `get_rewards_manager` function.
  - `src/storage.rs`: Utility functions to manage the contract storage.
  - `src/token.rs`: Provides the `create_contract` function.

- `liquidity_pool_swap_router` directory: `soroban-liquidity-pool-swap-router-contract` implementation. It calculates the swaps made by different exchanges and gives the better option.
    - `src/constants.rs`: `FEE_MULTIPLIER` definition.
    - `src/contract.rs`: Contract implementation.
    - `src/interface.rs`: Trait definitions for the contract.
    - `src/lib.rs`: Exports contract definitions.
    - `src/plane.rs`: pool plane client and related utilities.
    - `src/stableswap_pool.rs`: estimation procedure for stableswap pools.
    - `src/standard_pool.rs`: estimation procedure for standard pools.
    - `src/storage.rs`: Utility functions to get and set the plane in the storage.
- `rewards` directory: library used to calculate rewards in the pools.
    - `src/constants.rs`: `REWARD_PRECISION` constant definition.
    - `src/lib.rs`: `Rewards` and `RewardsConfig` structs implementation. It also exports all the library functionality.
    - `src/manager.rs`: Rewards manager implementation.
    - `src/storage.rs`: Utilities to persist reward-related data in the storage.
- `token_share` directory: Library used to interact with the `soroban-token-contract` defined in the token directory.
    - `src/lib.rs`: Full module implementation.
- `token` directory: `soroban-token-contract` implementation. It is a SEP-0041[1] compatible token smart contract designed for liquidity pool share management.
    - `src/allowance.rs`: Utility functions to authorize operations on behalf of other users.
    - `src/balance.rs`: Utility functions to manage a user's balance.
    - `src/contract.rs`: Contract implementation.
    - `src/lib.rs`: Exports contract client.
    - `src/metadata.rs`: Utility functions to handle the token metadata.
- `utils` directory: Library with utilities to handle storage and other minor things.
    - `src/bump.rs`: Utilities to keep data alive in the blockchain.
    - `src/constant.rs`: Library constants.
    - `src/lib.rs`: Exports functionality implemented in the library.
    - `src/storage.rs`: Macros to generate getters and setters backed by the soroban storage.
    - `src/utils.rs`: Assorted functions.

No other files in this repository were audited. Its dependencies are assumed to work according to their documentation. Also, no tests were reviewed for this audit.

[1] https://github.com/stellar/stellar-protocol/blob/master/ecosystem/sep-0041.md

Methodology

CoinFabrik was provided with the source code, including automated tests that define the expected behavior. Our auditors spent seven weeks auditing the source code provided, which includes understanding the context of use, analyzing the boundaries of the expected behavior of each contract and function, understanding the implementation by the development team (including dependencies beyond the scope to be audited) and identifying possible situations in which the code allows the caller to reach a state that exposes some vulnerability. Without being limited to them, the audit process included the following analyses.

- Arithmetic errors
- Race conditions
- Misuse of block timestamps
- Denial of service attacks
- Excessive gas usage
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Insufficient validation of the input parameters
- Incorrect handling of cryptographic signatures
- Centralization and upgradeability

The development team implemented the fixes and enhancements while the audit was conducted.

Findings

In the following table we summarize the security issues we found in this audit. The severity classification criteria and the status meaning are explained below. This table does not include the enhancements we suggest to implement, which are described in a specific section after the security issues.

ID	Title	Severity	Status
ME-01	Cannot Change Admin in soroban-token-contract	Medium	Resolved
ME-02	Minimum Minted Shares	Medium	Resolved

ID	Title	Severity	Status
MI-01	No-op admin setting in soroban-liquidity-pool-router-contract	Minor	Resolved
MI-02	Check Vector Sizes in soroban-liquidity-pool-contract	Minor	Resolved
MI-03	Repeated Trait Definitions	Minor	Acknowledged
MI-04	Denial of Service Via Custom Pools	Minor	Resolved
MI-05	`transfer_from` Usage	Minor	Resolved

Severity Classification

Security risks are classified as follows:

- **Critical:** These are issues that we manage to exploit. They compromise the system seriously. Blocking bugs are also included in this category. They must be fixed **immediately**.
- **High:** These refer to a vulnerability that, if exploited, could have a substantial impact, but requires a more extensive setup or effort compared to critical issues. These pose a significant risk and **demand immediate attention**.
- **Medium:** These are potentially exploitable issues. Even though we did not manage to exploit them or their impact is not clear, they might represent a security risk in the near future. We suggest fixing them **as soon as possible**.
- **Minor:** These issues represent problems that are relatively small or difficult to take advantage of, but might be exploited in combination with other issues. These kinds of issues do not block deployments in production environments. They should be taken into account and be fixed **when possible**.

Issues Status

An issue detected by this audit has one of the following statuses:

- **Unresolved**: The issue has not been resolved.

- **Acknowledged**: The issue remains in the code, but is a result of an intentional decision. The reported risk is accepted by the development team.
- **Resolved**: Adjusted program implementation to eliminate the risk.
- **Partially resolved**: Adjusted program implementation to eliminate part of the risk. The other part remains in the code, but is a result of an intentional decision.
- **Mitigated**: Implemented actions to minimize the impact or likelihood of the risk.

Critical Severity Issues

No issues found.

High Severity Issues

No issues found.

Medium Severity Issues

ME-01 Cannot Change Admin in soroban-token-contract

Found on commit: `8c481ea41c11f23db546d81071bb32d2763161d7`
Location:
- `token/src/contract.rs: 63`

The `Token.set_admin` function has the following signature (see line 56):

```
pub fn set_admin(e: Env, new_admin: Address) {
```

Yet when setting the new admin, instead of setting the `new_admin` it sets the old admin in line 63.

```
access_control.set_admin(&admin);
```

This leads to the admin to not change after the invocation.

Recommendation
Fix line 63 to set the new admin. In your tests, after setting the admin check that the new admin is being used when authorizing actions.

Status
Resolved. Fixed according to the recommendation. Checked on commit `85aab00b392170433eb82235e37a9160de4376b5`.

ME-02 Minimum Minted Shares

Found on commit: `aaa94a13cd3ff072788dc82a036d836787b54567`
Location:

- `liquidity_pool/src/contract.rs: 130`
- `liquidity_pool_router/src/contract.rs: 64`
- `liquidity_pool_stableswap/src/contract.rs: 800`

If the state of any of the pools changes, a user adding liquidity to the pool may get less shares than the ones they expected.

Recommendation

Add a `min_mint` parameter in the `deposit` functions of the pools and a `mint_mints` parameter in the router `deposit` function. Check in the pool's `deposit` functions that the number of shares minted comply with it.

Status

Resolved. The min_shares parameter was added to request a minimum number of shares. Checked on commit `02c79a24b0314abd5c52c454213c7bd3f152bad6`.

Minor Severity Issues

MI-01 No-op admin setting in `soroban-liquidity-pool-router-contract`

Found on commit: `85aab00b392170433eb82235e37a9160de4376b5`
Location:

- `liquidity_pool_router/src/contract.rs: 205-210`

If an admin is already set while calling the `init_admin` function, it will keep the original admin. This may lead to believing that the admin was changed even when it was not changed.

Recommendation

Fail when the admin was already set.

Status

Resolved. Checked on commit `afd7e3fb40c92fbeafb322e9bba7a6f3a29eba71`. The fix was also applied to a similar problem in `soroban-liquidity-pool-swap-router-contract`.

MI-02 Check Vector Sizes in `soroban-liquidity-pool-contract`

Found on commit: `afd7e3fb40c92fbeafb322e9bba7a6f3a29eba71`

Location:
- `liquidity_pool/src/contract.rs: 51, 75, 126, 312`

In the `initialize_all` and `initialize` functions it is not checked that the `tokens` vector size is 2. In the `deposit` function it is not checked that the `desired_amounts` vector size is 2. In the `withdraw` function it is not checked that the `min_amounts` vector size is 2. The lack of checks may lead to errors when using these functions, as this pool only supports 2 tokens.

Recommendation
Check that the size of the passed vectors is 2.

Status
Resolved. A similar issue was also resolved in the `calc_token_amount` function defined in `liquidity_pool_stableswap/src/contract.rs:83`. Checked on commit `aaa94a13cd3ff072788dc82a036d836787b54567`.

MI-03 Repeated Trait Definitions

Found on commit: `afd7e3fb40c92fbeafb322e9bba7a6f3a29eba71`

Some traits are defined in multiple places, but expected to have the same signatures.

For example see:

1. `Plane` (also named `PoolPlaneInterface`) defined in:
   a. `liquidity_pool/src/plane_interface.rs:3`.
   b. `liquidity_pool_router/src/pool_interface.rs:149`.
   c. `liquidity_pool_stableswap/src/plane_interface.rs:3`.
   d. `liquidity_pool_swap_router/src/interface.rs:3` (here it is named `RouterInterface` and has extra functions defined).
2. `LiquidityPoolCrunch` (also named `ManagedLiquidityPool`) defined in:
   a. `liquidity_pool/src/pool_interface.rs:3`.
   b. `liquidity_pool_stableswap/src/pool_interface.rs:3`.
3. `LiquidityPoolTrait` (also named `LiquidityPoolInterfaceTrait`) defined in:
   a. `liquidity_pool/src/pool_interface.rs:17`.
   b. `liquidity_pool_stableswap/src/pool_interface.rs:19`.
4. `UpgradeableContractTrait` (also named `UpgradeableContract`) defined in:
   a. `liquidity_pool/src/pool_interface.rs:74`.
   b. `liquidity_pool_router/src/router_interface.rs:3`.
   c. `liquidity_pool_stableswap/src/pool_interface.rs:82`.
   d. `liquidity_pool_swap_router/src/interface.rs:23`.
5. `RewardsTrait` defined in:
   a. `liquidity_pool/src/pool_interface.rs:82`.

b. `liquidity_pool_stableswap/src/pool_interface.rs:90`.

This is not an exhaustive list. Non-consistent definitions may lead to bugs, including security ones.

Recommendation

Look for all the trait definitions and consolidate them. All definitions shared between contracts should be made in a library imported by those contracts. Also make sure to mark all the structs that implement the trait as implementing the trait. This allows the compiler to check if the implementations respect the trait's interface definition.

Status

Acknowledged.

MI-04 Denial of Service Via Custom Pools

Found on commit: `afd7e3fb40c92fbeafb322e9bba7a6f3a29eba71`
Location:

- `liquidity_pool_swap_router/src/contract.rs:89`

When the `swap_routed` or `estimate_swap_routed` functions in the `soroban-liquidity-pool-router-contract` contract are invoked by a user, the `estimate_swap` function of the `soroban-liquidity-pool-swap-router-contract` is called. There the exchanged funds are estimated. This estimation is done only using the data published in the `soroban-liquidity-pool-plane-contract` contract. So if a custom pool is added to the `soroban-liquidity-pool-router-contract` contract via the `add_custom_pool` function and it is not a standard or stableswap contract, it will panic. This effectively means that pools that are not standard or stableswap are unsupported, even if they can be added. If they are effectively added, then they cannot be used and block the normal functionality for the token pair in the `soroban-liquidity-pool-router-contract` contract.

The severity of this issue was lowered because it requires an admin action (adding a custom pool) and it can be worked around by another admin action (removing the custom pool).

Recommendation

Remove the `soroban-liquidity-pool-plane-contract` and the `soroban-liquidity-pool-swap-router-contract` contracts. Delegate the swap estimation to the estimate_swap function of each pool contract in the `estimate_swap_routed` function of the `soroban-liquidity-pool-router-contract` contract[2].

Another option is to remove the possibility of adding custom pools in the `soroban-liquidity-pool-router-contract` contract altogether. If an extra pool type is required, the router contract can be upgraded to support it when needed.

Status

Resolved. Custom pool support has been removed. Checked on commit `02c79a24b0314abd5c52c454213c7bd3f152bad6`.

MI-05 `transfer_from` Usage

Found on commit: `aaa94a13cd3ff072788dc82a036d836787b54567`
Location:

- `liquidity_pool/src/contract.rs: 162, 168, 242, 339`
- `liquidity_pool_router/src/contract.rs: 368`
- `liquidity_pool_stableswap/src/contract.rs: 210, 272, 845, 930, 1012`
- `rewards/src/manager.rs: 138`

In soroban, it is a bad practice to use `transfer_from` to transfer tokens when using `transfer` can work, as the allowance mechanism is prone to errors such as leaving a pre-approved allowance either by error or to ease the use of the contract. This exposes the user to potential issues in the contract.

Recommendation

Use `transfer` instead of `transfer_from` to transfer funds. To better integrate with soroban's authorization framework, transfer all the funds and then transfer back all the unused funds if necessary.

Status

Resolved. All `transfer_from` calls were replaced by `transfer` calls. Checked on commit `de33577d289c008862a5b8b0bc561e6802c21ecc`.

Enhancements

These items do not represent a security risk. They are best practices that we suggest implementing.

[2] Given that calling the `soroban-liquidity-pool-swap-router-contract` and the `soroban-liquidity-pool-plane-contract` contracts is required for the current implementation. The recommended action is potentially cheaper.

ID	Title	Status
EN-01	Commented Code	Implemented
EN-02	Repeated Calculations	Not implemented
EN-03	Todo Comments	Implemented
EN-04	Swap Authorization Race Condition	Implemented
EN-05	Transfer Authorization Race Conditions in soroban-liquidity-pool-stableswap-contract	Implemented

EN-01 Commented Code

Suggested on commit: `85aab00b392170433eb82235e37a9160de4376b5`
Location:

- `liquidity_pool/src/contract.rs: 130, 150-151, 402-403`
- `liquidity_pool_stableswap/src/contract.rs: 349-351, 356-359, 412-414, 749, 898-900`
- `liquidity_pool_stableswap/src/token.rs: 9-10, 13-14`
- `liquidity_pool_swap_router/src/stableswap_pool.rs: 66-68, 73-76`
- `rewards/src/manager.rs: 98-99`

The location list is not exhaustive.

Recommendation

Remove commented code, as it reduces code readability.

Status

Implemented. Checked on commit `afd7e3fb40c92fbeafb322e9bba7a6f3a29eba71`.

EN-02 Repeated Calculations

Suggested on commit: `afd7e3fb40c92fbeafb322e9bba7a6f3a29eba71`
Location:

- `liquidity_pool/src/contract.rs`
- `liquidity_pool_stableswap/src/contract.rs`
- `liquidity_pool_swap_router/src/stableswap_pool.rs`
- `liquidity_pool_swap_router/src/standard_pool.rs`

The calculations used to compute the exchanged tokens in both `soroban-liquidity-pool-contract` and `soroban-liquidity-pool-stableswap-contract` are repeated in three places:

1. In the contract's `swap` function.
2. In the contract's `estimate_swap` function.
3. In `soroban-liquidity-pool-swap-router-contract`.

If for any reason the calculations differ, then the system will not be able to properly choose the best exchange to do the swap via the `swap_routed` function in the `soroban-liquidity-pool-router-contract` contract.

Recommendation

Extract all the common calculations to a library and invoke it in the three places.

Status

Not implemented.

EN-03 Todo Comments

Suggested on commit: `afd7e3fb40c92fbeafb322e9bba7a6f3a29eba71`
Location:

- `liquidity_pool/src/pool_interface.rs: 83`
- `liquidity_pool_stableswap/src/pool_interface.rs: 91`
- `rewards/src/manager.rs: 50, 308`

There are several places in the code where "to do" comments are present.

Recommendation

If the to do comment is correct, do it. If not, remove it.

Status

Implemented. All "to do" comments were removed. Checked on commit `02c79a24b0314abd5c52c454213c7bd3f152bad6`.

EN-04 Swap Authorization Race Condition

Suggested on commit: `aaa94a13cd3ff072788dc82a036d836787b54567`
Location:

- `liquidity_pool_router/src/contract.rs: 516, 532`

If the `estimate_swap_routed` invocation result in line 516 varies between the time when the transaction is built and when it is effectively executed in the network, the swap_routed function of `soroban-liquidity-pool-router-contract` will fail, as it will lack the proper authorization for the `swap` function invocation in line 532.

Recommendation

Transfer the funds to the router first and then from the router to the pool to avoid this issue, as the pool invocation will require the authorization of the router itself instead of the user.

Status

Implemented. The `swap_routed` function in `liquidity_pool_router/src/contract.rs` was removed, making the race condition impossible. Checked on commit `de33577d289c008862a5b8b0bc561e6802c21ecc`.

EN-05 Transfer Authorization Race Conditions in `soroban-liquidity-pool-stableswap-contract`

Suggested on commit: `de33577d289c008862a5b8b0bc561e6802c21ecc`
Location:

- `liquidity_pool_stableswap/src/contract.rs: 210, 842`

While fixing MI-05 transfer_from Usage, a new problem was introduced. The `deposit` and `liquidity_imbalance` functions may fail if the state of the blockchain changes between transaction building and transaction execution, as the amount of the `transfer` calls in lines 210 and 842 depends on the network state.

Recommendation

Transfer the full amount and then return back the difference, as implemented in the `deposit` function in `liquidity_pool/src/contract.rs`.

Status

Implemented. Checked on commit `f1f1f35d702b8283243f9bace37c1f84cd753950`.

Other Considerations

The considerations stated in this section are not right or wrong. We do not suggest any action to fix them. But we consider that they may be of interest to other stakeholders of the project, including users of the audited contracts, token holders or project investors.

Rewards

There is no code in the project to fund the rewards to be claimed for providing liquidity to any of the audited pools via the `claim` function. The development team informed us that they intend to manually fund the pools to have enough tokens to give up the rewards.

Centralization

The only fully decentralized contract analyzed in this audit is `soroban-liquidity-pool-plane-contract`.

In the rest of the contracts the admin has significant privileges and is a single point of failure. In the only non-upgradeable contract `soroban-token-contract`, the admin can mint an arbitrary amount of tokens for any user, but it must be noted that in this system the admin of the token is the corresponding liquidity pool. See `liquidity_pool/src/contract.rs:92` and `liquidity_pool_stableswap/src/contract.rs:751`[3].

Upgrades

`soroban-token-contract` cannot be upgraded by the admin. `soroban-liquidity-pool-plane-contract` also has no upgrade possibilities. The rest of the contracts can be upgraded by its admin.

Privileged Roles

These are the privileged roles that we identified on each of the audited contracts.

soroban-liquidity-pool-contract

admin

The account with the admin role can:

- set the rewards config via the `set_rewards_config` function.
- upgrade the contract via the `upgrade` function.

The initial admin is set via the `initialize` function.

The admin cannot be changed without a contract upgrade.

soroban-liquidity-pool-router-contract

admin

The account with the admin role can:

- upgrade the contract via the `upgrade` function.
- set or change the token hash, to be used in both of the pools initialization, via the `set_token_hash` function.

[3] In commit `85aab00b392170433eb82235e37a9160de4376b5`.

- set or change the constant-product pool hash, to be used in the pool deployment, via the `set_pool_hash` function.
- set or change the stableswap pool hash, to be used in the pool deployment, via the `set_stableswap_pool_hash` function.
- set or change the payment token, payment amount and payment address to be used in the initialization of the stableswap pool, via the `configure_init_pool_payment` function.
- set or change the rewards token to be used in the pool's initialization and the claim reward procedure via the set_reward_token function.
- set or change the rewards configuration for a pool via the `set_rewards_config` function.
- add a custom pool via the `add_custom_pool` function[4].
- remove a pool via the `remove_pool` function.
- set the pools plane via the `set_pools_plane` function.
- set the swap router via the `set_swap_router` function.

The initial admin is set via the `init_admin` function.

The admin cannot be changed without a contract upgrade.

soroban-liquidity-pool-stableswap-contract

admin

The account with the admin role can:

- ramp the a parameter via the `ramp_a` function.
- stop the ramping of the a parameter via the `stop_ramp_a` function.
- set the future fees via the `commit_new_fee` function.
- apply the new fee settings via the `apply_new_fee` function.
- cancel the change of fees via the `revert_new_parameters` function.
- set the future admin via the `commit_transfer_ownership` function.
- apply the new admin via the `apply_transfer_ownership` function.
- cancel the change of admin via the `revert_transfer_ownership` function.
- withdraw the admin fees via the `withdraw_admin_fees` function.
- add the admin fees to the reserves via the `donate_admin_fees`.
- after 2 months of operation, pause operations via the `kill_me` function.
- unpause operations via the `unkill_me` function.
- upgrade the contract via the `upgrade` function.
- set the rewards configuration via the set_rewards_config function.

The initial admin is set via the `initialize` function.

The admin cannot be changed without a contract upgrade.

[4] This functionality was removed to resolve MI-04 Denial of Service Via Custom Pools.

soroban-liquidity-pool-swap-router-contract

admin

The account with the admin role can:

- set the pools plane via the `set_pools_plane` function.
- upgrade the contract via the `upgrade` function.

The initial admin is set via the `init_admin` function.

The admin cannot be changed without a contract upgrade.

soroban-token-contract

admin

The account with the admin role can:

- mint tokens for an arbitrary address via the `mint` function.
- set a new admin via the `set_admin` function[5].

The initial admin is set via the `initialize` function.

Initialization

The analyzed contracts are expected to be initialized before use. As part of this audit, we assume that the contracts are properly initialized in the same transaction as the deploy occurs. These are the details for each one:

soroban-liquidity-pool-contract

It has to be initialized via the `initialize_all` function or, alternatively, it can be initialized by calling the `set_pools_plane`, `initialize` and `initialize_rewards_config` functions.

soroban-liquidity-pool-plane-contract

No initialization required.

soroban-liquidity-pool-router-contract

It has to be initialized via the `init_admin` function.

[5] See ME-01 Cannot Change Admin in soroban-token-contract.

soroban-liquidity-pool-stableswap-contract

It has to be initialized by calling the `set_pools_plane`, `initialize` and `initialize_rewards_config` functions.

soroban-liquidity-pool-swap-router-contract

It has to be initialized via the `init_admin` function.

soroban-token-contract

It has to be initialized via the `initialize` function.

Changelog

- 2024-02-02 – Reported ME-01.
- 2024-02-09 – Reported MI-01. Suggested EN-01. Checked fix for ME-01.
- 2024-02-16 – Reported MI-02, MI-03. Checked fix for MI-01. Checked implementation for EN-01.
- 2024-02-22 – Reported MI-04. Suggested EN-02, EN-03. Checked fix for MI-02.
- 2024-02-29 – Reported ME-02.
- 2024-03-06 – Reported MI-05. Suggested EN-04. Checked fixes for ME-02, MI-04. Checked implementation for EN-03.
- 2024-03-08 – Added Sections: Executive Summary, Scope, Methodology, Other Considerations. Suggested EN-05. Checked fix for MI-05. Registered acknowledgement for MI-03. Checked implementation for EN-04.
- 2024-03-25 – Checked implementation for EN-05.

Disclaimer: This audit report is not a security warranty, investment advice, or an approval of the Aquarius project since CoinFabrik has not reviewed its platform. Moreover, it does not provide a smart contract code faultlessness guarantee.