CALC Finance Audit Report

Prepared for CALC Finance, 9th November 2022

Table of Contents

Introduction

SCV was engaged by CALC Finance to assist in identifying security threats and vulnerabilities that have the potential to affect their security posture. Additionally, SCV will assist the team in understanding the risks and identifying potential mitigations.

Scope

SCV performed the security assessment on the following codebase:

- https://github.com/CALC-FINANCE/calc-rs
- Code Freeze: `e81f8da4e29ba78940cf64a7c84b94ffa0256052`

Remediations were applied in several PRs and were reviewed by SCV up to the following:

- https://github.com/CALC-FINANCE/calc-rs/pull/192

Methodologies

SCV performs a combination of automated and manual security testing based on the scope of testing. The testing performed is based on the extensive experience and knowledge of the auditor to provide the greatest coverage and value to CALC Finance. Testing includes, but is not limited to, the following:

- Understanding the application and its code base purpose;
- Deploying SCV in-house tooling to automate dependency analysis and static code review;
- Analyse each line of the code base and inspect application security perimeter;
- Review underlying infrastructure technologies and supply chain security posture;

Code Criteria and Test Coverage

SCV used a scale from 0 to 10 that represents how *SUFFICIENT* or *NOT SUFFICIENT* each code criteria was during the assessment:

Criteria	Status	Notes
Provided Documentation	SUFFICIENT 🟢	N/A
Code Coverage Test	SUFFICIENT 🟢	N/A
Code Readability	SUFFICIENT 🟢	N/A
Code Complexity	SUFFICIENT 🟢	N/A

Audit Observations

1. The implementation between the Kujira Fin contract and the contract is out of scope due to the Kujira Fin contract's source code is not public.

2. The contract relies on an executor to execute the permissionless `ExecuteTrigger` functionality to trigger configurations.

Vulnerabilities Summary

#	Summary Title	Risk Impact	Status
1	DCA contract migration would cause pending funds stuck in the contract	**Severe**	**Resolved**
2	Anyone can cancel vaults and deposit funds on behalf of the vault owner	**Severe**	**Resolved**
3	Incorrect accounting after retracting limit order	**Severe**	**Resolved**
4	Incorrect recipient address set	**Medium**	**Resolved**
5	A large time gap would cause time interval recovery to fail	**Medium**	**Resolved**
6	CACHE should be saved for sub-message execution	**Medium**	**Resolved**
7	Querying a vault's position type can return an incorrect value	**Low**	**Resolved**
8	Triggers should be removed upon successful vault cancellations	**Low**	**Resolved**
9	Vault started_at time can be forcefully configured as None	**Low**	**Resolved**
10	The swap amount should be higher than 0	**Low**	**Resolved**
11	Consider implementing paginations for queries	**Low**	**Partially resolved**
12	Validator address validation can be bypassed	**Low**	**Acknowledged**
13	User tokens are not automatically delegated upon sub-message failure	**Informational**	**Acknowledged**
14	Duplicate events are stored after retracting limit order	**Informational**	**Resolved**
15	Unneeded storage operations	**Informational**	**Resolved**
16	Contract version is not updated after migration	**Informational**	**Resolved**
17	Destination's allocation amount can be 0	**Informational**	**Resolved**

18	General address validations	**Informational**	**Resolved**
19	GetVault query argument can be reduced	**Informational**	**Resolved**

Detailed Vulnerabilities

1 – DCA contract migration would cause pending funds stuck in the contract

---

Risk Impact: Severe - **Status**: Resolved

Description

The `migrate` functionality in `contracts/dca/src/contract.rs:46-48` attempts to remove all vaults, triggers, and events in the contract. As the storage state records how many balances are available for a specific user, a successful migration would cause the records to be wiped out. If any outstanding balance can be withdrawn from the user, they won't be able to withdraw anymore.

Recommendations

Consider reworking the implementation to allow users to withdraw their outstanding balance in a separate storage state.

Notes

This issue is resolved in commit `a473a01a23cf31a1b95159342b081f044ba28413`. The migration functionality will be empty when the proposal is submitted. They will implement the migration logic during the time of a contract upgrade.

2 − Anyone can cancel vaults and deposit funds on behalf of the vault owner

Risk Impact: Severe - **Status**: Resolved

Description

The `cancel_vault` and `deposit` functionalities in `contracts/dca/src/handlers/cancel_vault.rs:29` and `contracts/dca/src/handlers/deposit.rs:28` checks authentication by verifying the provided address argument equals to the vault owner. A malicious actor can bypass the defense by simply querying the vault owner and providing it as the address argument.

Recommendations

Consider checking whether the `info.sender` is the vault owner when performing `cancel_vault` and `deposit` functionalities.

Notes

The issue is resolved in commit `b7ae1b38203d7b80785917eb802d847723bf437f`. The team decided to leave the deposit functionality unguarded to enable composability where some other smart contract may deposit funds to a user's wallet on their behalf. Additionally, this can safeguard against depositing into an incorrect vault, as the sender must ensure the address to which they send tokens is correct.

3 – Incorrect accounting after retracting limit order

Risk Impact: Severe - **Status**: Resolved

Description

In `contracts/dca/src/handlers/after_fin_limit_order_retracted.rs:50`, `vault.get_swap_amount().amount` should be used instead of the `vault.swap_amount`. If the user's balance is lower than the swap amount, their balance would be used to swap on Fin instead of the swap amount (see `contracts/dca/src/handlers/create_vault.rs:178`).

Recommendations

Consider modifying line 44 into `vault.balance.amount - (vault.get_swap_amount().amount - amount_retracted)`.

Notes

The issue is resolved in commit `e5fbbdd84bbc6b8b450a31da4f34e7324dcd1aae`.

4 – Incorrect recipient address set

Risk Impact: Medium - **Status**: Resolved

Description

In `contracts/dca/src/handlers/create_vault.rs:54`, if the caller provides no destinations, the contract will automatically transfer all the funds back to them. This is incorrect as the caller can provide an optional vault owner as seen in `contracts/dca/src/contract.rs:130`. As a result, the funds that should be transferred to the vault owner are incorrectly transferred to the caller.

For example, if a smart contract performs vault creation execution with its users as the vault owner, the funds will be transferred back to the smart contract and not their intended users.

Recommendations

Consider modifying the destination address recipient to be the vault `owner` instead of `info.sender`.

Notes

The issue is resolved in commit `bbff02d86338846498234a370f3299a895ddc614`.

5 – A large time gap would cause time interval recovery to fail

Risk Impact: Medium - **Status**: Resolved

Description

The `get_next_target_time` functionality in `packages/base/src/helpers/time_helpers.rs:25-44` would try to recover the time interval if the current time is larger than the next execution time. This is done by limiting the duration interval until the next execution time is in the future of the current time.

The execution might fail if there is a large time gap between the current time and the next execution time due to an out-of-gas error. For instance, if the gap is seven days apart, the `recover_hourly_interval` function would need to loop 168 times to successfully set the next execution time (7 days * 24 hours). This means the execution must be called regularly to prevent out-of-gas failures. In accidental events such as a chain halt or off-chain bot failures, the issue might be introduced and cause the function to fail.

Recommendations

Consider reworking the implementation to calculate the difference in the duration and divide it by the interval to get the total number of intervals that needs to be recovered. After that, the total number of intervals can be added to the current time with an additional one interval to make the next execution time larger than the present time.

Notes

The issue is resolved in PR #152.

6 – CACHE should be saved for sub–message execution

Risk Impact: Medium - **Status**: Resolved

Description

In `calc-rs/contracts/dca/src/handlers/execute_trigger.rs:122` and line `127`, the `AFTER_FIN_SWAP_REPLY_ID` reply identifier will be triggered and causing the sub-message to execute `after_fin_swap` functionality. However, the `CACHE` storage state is not saved to hold the vault identifier and owner value. As a result, the execution would fail in `calc-rs/contracts/dca/src/handlers/after_fin_swap.rs:23`.

Recommendations

Consider saving the vault's identifier and owner value during `execute_trigger` functionality.

Notes

This issue was independently discovered by the team.

7 – Querying a vault's position type can return an incorrect value

Risk Impact: Low - **Status**: Resolved

Description

When creating a vault, a user can provide an optional position type as seen in `contracts/dca/src/handlers/create_vault.rs:32`. However, during execution the position type is determined by the pair and the balance denom (see `contracts/dca/src/handlers/execute_trigger.rs:28`). This is problematic because the `get_vault` functionality returns the vault state from the storage which includes the vault position provided by the user.

For example, suppose the user decides to send USDC for a pair that consists of KUJI/USDC. In that case, this means the user's position should be "`PositionType::Enter`" (`contracts/dca/src/vault.rs:48`). If the user provided the position type as "`PositionType::Exit`", the query functionalities will reflect that although the real position is the other way around.

Recommendations

Consider removing the option for the user to provide the position type and instead dynamically determine the position type using the `get_position_type` functionality in query messages.

Notes

This issue was resolved in commit `fc135a937635e9533e6c5956b1280f8575a3beb9`.

8 – Triggers should be removed upon successful vault cancellations

Risk Impact: Low - **Status**: Resolved

Description

In `contracts/dca/src/handlers/after_fin_limit_order_retracted.rs:70-100`, `delete_trigger` is not called to remove the vault identifier and trigger configuration from the `TRIGGERS` and `TRIGGER_ID_BY_FIN_LIMIT_ORDER_IDX` storage state. As the order is retracted successfully, the trigger storage state should also be removed.

Recommendations

Consider calling `delete_trigger` in `contracts/dca/src/handlers/after_fin_limit_order_retracted.rs:70-100`.

Notes

This issue was resolved in commit `f2fe964e004725e758ba857a70a2ab7000aa4160`.

9 – Vault started_at time can be forcefully configured as None

Risk Impact: Low - **Status**: Resolved

Description

In `contracts/dca/src/handlers/deposit.rs:47-48`, if the vault's balance after deposit is higher than the swap amount, the vault's status will become active. On the other end, the vault's `started_at` value would only be set if the vault status is scheduled (see `contracts/dca/src/handlers/execute_trigger.rs:39`). This is problematic because if a user deposits more funds to their vault before executing the trigger, the vault's `started_at` time will reflect `None` permanently.

Recommendations

Consider performing an additional check whether the vault's `started_at` value is `None` in `contracts/dca/src/handlers/execute_trigger.rs:35`. For example, `if vault.is_scheduled() || vault.started_at == None`.

Notes

This issue is resolved in commit `1f23da32e03655aee771a9503082b1c74aa58da2`.

10 – The swap amount should be higher than 0

Risk Impact: Low - **Status**: Resolved

Description

When creating a vault in the DCA contract, the swap amount is only validated to be lesser or equal to the sent balance via the `assert_swap_amount_is_less_than_or_equal_to_balance` functionality. This means that the swap amount can be configured as 0. As a result, a user that chooses `TriggerConfiguration::Time` as trigger execution will cause their execution to fail in `contracts/dca/src/handlers/execute_trigger.rs:112` or line `117`.

Recommendations

Consider verifying the swap amount is not 0 in `assert_swap_amount_is_less_than_or_equal_to_balance`.

Notes

The issue is resolved in commit `7bcf78b457d6184c45ef2bee35f7bdaeb153a7c1`.

11 – Consider implementing paginations for queries

Risk Impact: Low - **Status**: Partially resolved

Description

In the DCA contract, some queries do not implement paginations and instead simply perform an unbounded query operation.

Affected functionalities and code lines:

- `get_events_by_resource_id` functionality would fail if too many events are recorded for the specific vault identifier
  - `contracts/dca/src/handlers/get_events_by_resource_id.rs:11`
- `get_pairs` functionality would fail if too many `PAIRS` instantiated
  - `contracts/dca/src/handlers/get_pairs.rs:7`
- `get_time_trigger_ids` functionality would fail if too many triggers configured in the past
  - `contracts/dca/src/handlers/get_time_trigger_ids.rs:8-13`

Recommendations

Consider implementing paginations for the affected queries.

Notes

This issue is partially resolved in commit `6544819cfefa6df1970d7bb06e206b0f822a74df` and `973a1d13e3e32242c5ecd2c709a5396e813ea25`. Additionally, the team mentioned that the `get_pairs` functionality is admin-controlled, so implementing a pagination mechanism would be unneeded.

12 – Validator address validation can be bypassed

Risk Impact: Low - **Status**: Acknowledged

Description

The `assert_validator_is_valid` functionality in `contracts/dca/src/validation_helpers.rs:218` simply checks whether the validator contract implements the `StakingQuery::Validator` query message. A malicious contract can bypass this defence by implementing the required query message and returning a valid `ValidatorResponse` response.

We consider this issue with a low-risk impact because the user determines destinations.

Recommendations

Consider whitelisting specific validators that users can delegate to.

Notes

The team mentioned they do not wish to maintain a list of validators.

13 – User tokens are not automatically delegated upon sub–message failure

Risk Impact: Informational - **Status**: Acknowledged

Description

Sub-message executions are always handled using `ReplyOn::Always` to continue execution even when a failure is encountered. Instead of reverting the whole transaction, the error is handled gracefully while the previous state changes are committed. However, one of the failures will cause the delegation functionality to fail temporarily.

In `contracts/dca/src/handlers/after_z_delegation.rs:53-58`, a `DCAVaultDelegationFailed` event is recorded when the sub-message execution fails. However, the transaction is not reverted due to line `61` returning the response as success. This might happen if the staking router admin hasn't added the DCA contract address as an allowed caller using the `AddAllowedZCaller` message.

Since the delegation message is executed after a filled limit order or a successful `Fin` swap in `contracts/dca/src/handlers/after_fin_limit_order_withdrawn_for_execute_trigger.rs:134` and `contracts/dca/src/handlers/after_fin_swap.rs:114`, previous state changes are committed hence users are unable to replay the action unless they are willing to swap back KUJI into the other denom which their limit order or swap is executed before.

As a result, the user's funds will not automatically be delegated to the desired delegator, and they must manually delegate their tokens, which defeats the purpose of the DCA contract.

Recommendations

Consider gracefully handling the failure by automatically performing delegation on the next interval batch.

Notes

The team mentioned that this is an intentional design decision to ensure swaps still succeed even if the delegation failed for some reason. They will capture these failed events and surface them to the user interface so post actions such as canceling or reauthorizing can be performed.

14 – Duplicate events are stored after retracting limit order

Risk Impact: Informational - **Status**: Resolved

Description

In `contracts/dca/src/handlers/after_fin_limit_order_retracted.rs:41-44`, a `DCAVaultCancelled` event data is created for the specific vault identifier. However, this event is already stored when the user attempts to cancel the vault as seen in `contracts/dca/src/handlers/cancel_vault.rs:32-35`. This would cause duplicate events to be stored in the contract storage.

Recommendations

Consider removing `contracts/dca/src/handlers/after_fin_limit_order_retracted.rs:41-44`.

Notes

This issue was resolved in commit `a589bc06e3e7927ec20d05760a3825908086a79a`.

15 – Unneeded storage operations

Risk Impact: Informational - **Status**: Resolved

Description

In several instances of the codebase, there are occurrences where the storage is removed before saving. As CosmWasm storage will be overwritten by default when saving items, some of the removed operations can be eliminated to reduce computation power and gas usage.

Affected code lines:

- `contracts/dca/src/contract.rs:49`
- `contracts/dca/src/handlers/after_fin_limit_order_submitted.rs:27`
- `contracts/staking-router/src/contract.rs:19`

Recommendations

Consider removing the above lines.

Notes

This issue is resolved in commit `00c5915731b0fde2cd879a676c8b2914539814eb`. However, the DCA and staking router contract's migration entry points are not removed due to the migration logic will be implemented after the submitted proposal.

16 – Contract version is not updated after migration

Risk Impact: Informational - **Status**: Resolved

Description

During contract initialization, `set_contract_version` functionality is called to store the contract name and version. This should also be performed to update the latest contract name and version.

Affected migration entry points:

- `contracts/dca/src/contract.rs:45`
- `contracts/staking-router/src/contract.rs:18`

Recommendations

Consider updating the contract name and version during migrations.

Notes

This issue is resolved in commit `a473a01a23cf31a1b95159342b081f044ba28413`. The team mentioned that they would implement the migration logic during the contract upgrade as the migration function will be empty when submitting the proposal.

17 – Destination's allocation amount can be 0

Risk Impact: Informational - **Status**: Resolved

Description

The `assert_destination_allocations_add_up_to_one` functionality in `contracts/dca/src/validation_helpers.rs:188` validates the total destination's allocation sum to be 100. There is a possibility that the allocation's amount is 0, which consumes unnecessary computation power and gas consumption.

Recommendations

Consider validating the destination's allocation amount is not 0.

Notes

The issue was fixed in commit `bd0a218c386f2dd42f2321ec86b813767ba7d495`.

18 – General address validations

Risk Impact: Informational - **Status**: Resolved

Description

In several instances of the codebase, input arguments are `Addr` variable type, but their values are not validated.

Affected code lines:

- `contracts/dca/src/handlers/update_config.rs:23`
- `contracts/dca/src/handlers/update_config.rs:25-26`
- `contracts/staking-router/src/contract.rs:41`
- `contracts/staking-router/src/contract.rs:42`
- `contracts/staking-router/src/contract.rs:23`
- `contracts/staking-router/src/contract.rs:24`

Recommendations

Consider validating the address value to be valid.

Notes

The issue is resolved in commit `eac10e239ebb6ae5a7c0292753d24478b9c041f1` and `817587a6bc83033359b6c846495b03cbdfacf163`.

19 – GetVault query argument can be reduced

Risk Impact: Informational - **Status**: Resolved

Description

The `get_vault` functionality in `contracts/dca/src/handlers/get_vault.rs:9` checks whether the provided address argument equals to the vault owner. As the vault owner value can be retrieved directly from public transactions, this check can be bypassed easily.

Recommendations

Consider removing the address argument from the query and simply return the `VaultResponse` based on the `vault_id` to be queried.

Notes

This issue is resolved in commit `5fede880439873ea6224e09b9b5b95e4011f50af`.

Document control

Version	Date	Approved by	Changes
0.1	04/11/2022	Vinicius Marino	Document Pre-Release
0.2	04/11 - 09/11	SCV Team	Remediation Revisions
1.0	09/11/2022	Vinicius Marino	Final Document Release

Appendices

A. Appendix – Risk assessment methodology

A qualitative risk assessment is performed on each vulnerability to determine the impact and likelihood of each.

Risk rate will be calculated on a scale. As per criteria Likelihood vs Impact table below:

	Rare	Unlikely	Possible	Likely
Critical	**Medium**	**Severe**	**Critical**	**Critical**
Severe	**Low**	**Medium**	**Severe**	**Severe**
Moderate	**Low**	**Medium**	**Medium**	**Severe**
Low	**Low**	**Low**	**Low**	**Medium**
Informational	**Informational**	**Informational**	**Informational**	**Informational**

LIKELIHOOD

- Likely: likely a security incident will occur;
- Possible: It is possible a security incident can occur;
- Unlikely: Low probability a security incident will occur;
- Rare: In rare situations, a security incident can occur;

IMPACT

- Critical: May cause a significant and critical impact;
- Severe: May cause a severe impact;
- Moderate: May cause a moderated impact;
- Low: May cause low or none impact;
- Informational: May cause very low impact or none.

B. Appendix – Report Disclaimer

The content of this audit report is provided “As is”, without representations and warranties of any kind.

The author and their employer disclaim any liability for damage arising out of, or in connection with, this audit report.

Copyright of this report remains with the author.