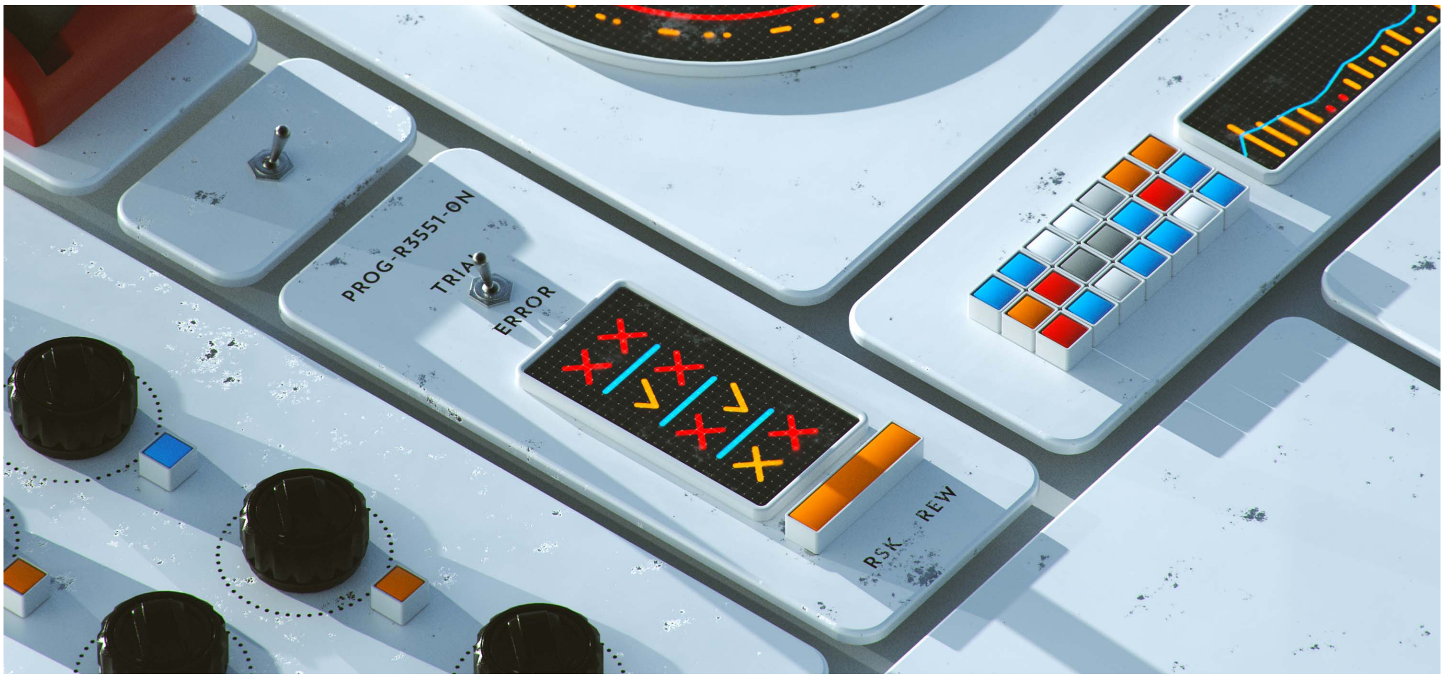

DCA Audit Report

---

title: DCA Clarity Smart Contract Audit Report
author: Nolan(X:@ma1fan)
date: October 25, 2024

Table of Contents

- Low
- Informational

Risk Classification

	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Summary

Project Name	**DCA Clarity Smart Contract Audit Report**
Repository	**https://github.com/cbadawi/dcahq-contracts**
Commit	**3ad8d06c7bf1c878b9c399f5054ab5e7bf8a5664**
Audit Timeline	**Oct 3 - Oct 20 th**
Methods	**Manual Review, Security Testing**

Issues Found

	Count
Critical Risk	4
High Risk	0
Medium Risk	2
Low Risk	2
Informational	1
Total Issues	**9**

Summary of Findings

ID	Description	Status
C-1	function `set-sources-targets-config` needs to rewrite	Resolved
C-2	only has `set-sources-targets-config` function to insert the data to `sources-targets-config` map	Resolved
C-3	It should add a function which can remove the treasury address	Resolved
C-4	price fetch may fail with function get-price-internal	Resolved
M-1	total-source-amount may be equal to fee, so it will lead to `traded-source-amount` may be 0, so in here we should check whether it is equal or not.	Resolved
M-2	`get-price-b` may be return 0, should check it in the function `dca-users-b`	Resolved
L-1	In the function `dca-users-b` , It should check `div-down-6` the second parameter is zero or not	Resolved
L-2	In `get-price-b` here we should check `amt-target` is 0 or not.	Resolved
I-1	In the function `dca-users-b` . It should change the assert sequence. We should firstly check the authorized or not	Acknowledge

Findings

Critical

C-1 function `set-sources-targets-config` needs to rewrite:

1. It should make sure `min-dca-threshold` is smaller than `max-dca-threshold` ,if not may lead to unexpected situation
2. It should check if the source principal is equal to target principal or not, if equal should revert the function

```
(define-public (set-sources-targets-config (source principal)

                                            (target principal)

                                            (id uint)

                                            (fee-fixed uint)

                                            (fee-percent uint)

                                            (source-factor uint)

                                            (helper-factor uint)

                                            (is-source-numerator bool)

                                            (min-dca-threshold uint)

                                            (max-dca-threshold uint)

                                            (max-slippage uint)

```

```
                    (token0 principal)
                    (token1 principal)
                    (token-in principal)
                    (token-out principal)
                    )
        (let ((value {id:id, fee-fixed:fee-fixed, fee-percent:fee-percent, source-factor: source-factor, helper-factor:helper-factor, is-source-numerator:is-source-numerator, min-dca-threshold: min-dca-threshold, max-dca-threshold: max-dca-threshold, max-slippage: max-slippage, token0: token0, token1: token1, token-in: token-in, token-out: token-out}))
                (asserts! (is-approved) ERR-NOT-AUTHORIZED)
                (print {function:"set-sources-targets-config", params: value, source:source, target:target})
                (ok (map-set sources-targets-config {source: source, target: target} value))
))
```

C-2 only has `set-sources-targets-config` function to insert the data to `sources-targets-config` map

In the contract, no function to remove the `sources-targets-config` data, if inserting data is outdated or useless, may lead to unexpected situation

So in the contract we should create a function named such as `remove-sources-targets-config` to remote the special data if the data is useless

C-3 It should add a function which can remove the treasury address

```
(define-public (set-treasury (address principal))
    (begin
    (asserts! (is-approved) ERR-NOT-AUTHORIZED)
    (ok (var-set treasury address))
))
```

C-4 price fetch fail with function get-price-internal

In this function . if in here target is not equal to `SP102V8P0F7JX67ARQ77WEA3D3CFB5XW39REDT0AM.token-wstx-v0-0` will lead to seriouse problem.when neither of the tokens is wstx, such as Alex-LiAlex pool,the price fetch will fail with err 2001.

```
(define-private (get-price-internal (source principal) (target principal) (factor uint))
        (let ((token-x (if (is-eq target 'SP102V8P0F7JX67ARQ77WEA3D3CFB5XW39REDT0AM.token-wstx-v3-0) target source))
                    (token-y (if (is-eq target 'SP102V8P0F7JX67ARQ77WEA3D3CFB5XW39REDT0AM.token-wstx-v3-0) source target))
                    )
        (contract-call? 'SP102V8P0F7JX67ARQ77WEA3D3CFB5XW39REDT0AM.amm-pool-v2-01 get-price token-x token-y factor)
))
```

Medium

M-1 total-source-amount may be equal to fee, so it will lead to `traded-source-amount` may be 0, so in here we should check whether it is equal or not.

The fee in here calcuated by `user-amounts` and `(get fee-fixed source-target-config)` which the Maclious user can control the user-amounts and make the multiple value fee equal to `traded-source-amount`

```
                                                                (let ((fee (* (get fee-fixed source-target-config) (len (filter is-none-zero user-amounts))))
(traded-source-amount (- total-source-amount fee))
                                                                        )
                                                                        (add-fee fee source)
```

M-2 `get-price-b` may be return 0, should check it in the function `dca-users-b`

The price value line 13 can be 0 by call the function `get-price-b` if price is 0, which will lead to the amt-in is 0 ,when call function `velar-swap-wrapper` .so we should check if the price is 0 or not ,it is 0 It should return error.

```
                                                          (let ((total-target-
amount (as-contract (try! (contract-call? dca-strategy velar-swap-wrapper id
                                      token0
                                      token1
                                      token-in
                                      token-out
                                      share-fee-to
                                      traded-source-amount
```

```
                                        (mul-down-6 (if is-source-numerator
                       (mul-down-6 price traded-source-amount)
                       (div-down-6 traded-source-amount price))
               (- ONE_6 max-slippage)) ))))
```

Low

L-1 In the function `dca-users-b` , It should check `div-down-6` the second parameter is zero or not

In the function `dca-users-b` no check the `price` is 0 or not . Here, the price may be 0(The price returned by `agg-amounts` may be 0), so we should check the price if it is 0, return error

```
                (unwrap! (map-get? approved-startegies (contract-of dca-
strategy)) ERR-INVALID-STRATEGY)
                    (print {user-amounts: user-amounts})
                    (let ((agg-amounts (fold aggregate-amounts user-amounts
{total-amount: u0, fee: u0, price: u0}))
```

```
                            (source-total-amount (get total-amount agg-amounts))
                            (fee (get fee agg-amounts))
                            (id (get id source-target-config))
                            (max-slippage (get max-slippage source-target-config))
                            (is-source-numerator (get is-source-numerator source-target-config))
                            (price (get price agg-amounts))
                            (amount-dy (if is-source-numerator (mul-down-6 price source-total-amount) (div-down-6 source-total-amount price))) ;; u12_058693
                            (min-dy (mul-down-6 amount-dy (- ONE_6 max-slippage)))
                        )
```

L-2 In `get-price-b` here we should check `amt-target` is 0 or not.

```
 (define-read-only (get-price-b (id uint) (token0 principal) (token-in principal) (amt-source uint) (is-source-numerator bool))
    (let ((pool (contract-call? 'SP1Y5YSTAHZ88XYK1VPDH24GY0HPX5J4JECTMY4A1.univ2-core do-get-pool id))
                (is-token0 (is-eq token0 token-in))
                (amt-target  (try! (contract-call? 'SP1Y5YSTAHZ88XYK1VPDH24GY0HPX5J4JECTMY4A1.univ2-library get-amount-out
                                                        amt-source
                                                        (if is-token0 (get reserve0 pool) (get reserve1 pool)) ;; reserve-in
                                                        (if is-token0 (get reserve1 pool) (get reserve0 pool)) ;; reserve-out
                                                        (get swap-fee pool) )))
                (price (if is-source-numerator (div-down-6 amt-target amt-source) (div-down-6 amt-source amt-target)))
            )
            (print {function: "get-price-b", input:{id: id, token0: token0, token-in: token-in, amt-source:amt-source},
                                                    more: {price: price,  pool: pool, amt-target: amt-target}})
            (ok price)
))
```

Informational

I-1 In the function `dca-users-b`. It should change the assert sequence. We should firstly check the authorized or not

For now

```
      (asserts! (> total-source-amount u0) ERR-INVALID-AMOUNT)
      (asserts! (and (>= mock-price (mul-down-6 price (- ONE_6 max-slippage))) (<= mock-price (mul-down-6 price (+ ONE_6 max-slippage)))) ERR-INVALID-PRICE)
       (asserts! (is-approved-dca-network) ERR-NOT-AUTHORIZED)
```

Should change the sequence to

```
     (asserts! (is-approved-dca-network) ERR-NOT-AUTHORIZED)
    (asserts! (> total-source-amount u0) ERR-INVALID-AMOUNT)
     (asserts! (and (>= mock-price (mul-down-6 price (- ONE_6 max-slippage))) (<= mock-price (mul-down-6 price (+ ONE_6 max-slippage)))) ERR-INVALID-PRICE)

```