ZER0F

Contract Security Analysis

Project	Arkadiko
Date	September 2021
Git revision	#3a3d660

Overview

The Arkadiko team asked us to review their Protocol smart contract. We looked at the code and now publish the result. In general, the scope of this project is focused on the onchain components of the protocol.

Per ZeroF demand, the Arkadiko team provided a list of actors and actor stories. The present security analysis is based on this resource.

User

Users can use the protocol in a normal way without having access to revenue. By default, anyone interacting with the protocol is a user.

Stories

Epic	Story
Vault	As a user, I can create a vault with collateral type STX-A or STX-B to mint USDA
	As a user, I can deposit extra STX in my vaults to improve my collateral to debt ratio
	As a user, I can withdraw STX from my vaults, which lowers my collateral to debt ratio
	As a user, I can withdraw STX if and only if they are not stacked in PoX.
	As a user, I can signal to the protocol that I no longer want my STX tokens to be stacked in PoX.
	As a user, I can swap two tokens with default or adjusted slippage.
	As a user, I can provide liquidity on a pair
	As a user, I can remove liquidity on a pair
	As a user, I can launch a governance vote if I have >=1% of the DIKO token supply
	As a user, I can signal to the protocol that a vault is unhealthy (i.e. too low collateralization levels)
	As a user, I can burn debt in a vault
	As a user, I can pay my stability fees on a vault
	As a user, I can stake single-asset DIKO or LP tokens which gives me stDIKO
Staking	As a user, I can unstake DIKO or LP tokens which burns stDIKO
	As a user, I can check my pending DIKO rewards through the staking module
	As a user, I can claim my pending DIKO rewards
	As a user, I can stake single-asset DIKO which gives me stDIKO
	As a user, I can start a 10 day cooldown period to unstake DIKO
Stacking	As a user, I can unstake DIKO within 2 days after the cooldown period which burns stDIKO
	As a user, I can stake LP tokens which claims my rewards
	As a user, I can unstake LP tokens at any time and rewards will be claimed
	As a user, I can check and claim DIKO rewards from staked LP tokens
	As a user, I can use the emergency-withdraw method to get LP tokens back without rewards
Auction	As a user, I can place a bid on a lot in an auction of a liquidated vault
	As a user, I can buy a lot by bidding the maximum
	As a user, I can redeem the collateral of a lot once auction ends
Governance	As a user, I can add a new proposal to governance
	As a user, I can vote for or against a proposal using DIKO or stDIKO
	As a user, I can end and execute a proposal

DAO Admin

Owns admin keys to the DAO. Can never withdraw LTV or any of the user's tokens. Can only withdraw revenue that the protocol generates. There can be one DAO admin which can be changed by the admin, calling a new funcDAO Admintion changing the address.

Stories

Story
As a DAO admin, I can withdraw revenue from the Freddie smart contract.
As a DAO admin, I can withdraw revenue from the auction engine smart contract.
As a DAO admin, I cannot toggle the emergency switches.
As a DAO admin, I can start stacking through stacker-1, stacker-2, stacker-3 and stacker-4 contracts
As a DAO admin, I can run the payout scripts for PoX
As a DAO admin, I can do everything a user can.

Guardian

Special private keys that can enable an emergency switch. There can be one guardian which can be changed by the guardian, calling a new function changing the address.

Stories

Story
As a guardian, I can toggle emergency switches on all smart contracts
As a guardian, I can do everything a user can.

Notes

This review was time-boxed to 10 days total. This security analysis is exclusively focused on the Clarity contracts and do not include an analysis of the protocol itself and offchain components eventually required by the Arkadiko protocol.

Findings

Severity	Finding
High	`0f-ARK-001` `0f-ARK-003` `0f-ARK-006` `0f-ARK-007` `0f-ARK-008`
Medium	`0f-ARK-005`
Low	`0f-ARK-004` `0f-ARK-011` `0f-ARK-014`
Informational	`0f-ARK-002` `0f-ARK-009` `0f-ARK-010` `0f-ARK-012` `0f-ARK-013` `0f-ARK-015`

0f-ARK-001: Ability to drain token STX and SIP10 reserves by manipulating vaults

Type	**high severity**
Git commit	https://github.com/arkadiko-dao/arkadiko/commit/3a3d66012041da0cd386d16d7360aa863d38f25c

Context

This issue is tracking a class of bugs that are sharing a root cause: the contract `arkadiko-freddie-v1-1` is blindly trusting its `reserve` arguments. A good amount of exploits can be derived from this flaw.

Demonstration

The entrypoint for creating vaults, `arkadiko-freddie-v1-1::collateralize-and-mint` has the following signature:

```
(define-public (collateralize-and-mint
  (collateral-amount uint)
  (debt uint)
  (pox-settings (tuple (stack-pox bool) (auto-payoff bool)))
  (collateral-type (string-ascii 12))
  (reserve <vault-trait>)
  (ft <ft-trait>)
  (coll-type <collateral-types-trait>)
  (oracle <oracle-trait>)
)
```

In this function, the argument `reserve` is being blindly trusted and used. A malicious user could pass a contract with an implementation for this trait of their own:

```
(define-public (collateralize-and-mint
  (token <ft-trait>)
  (token-string (string-ascii 12))
  (ucollateral-amount uint)
  (debt uint)
  (sender principal)
  (stacker-name (string-ascii 256))
  (stack-pox bool)
)
    (ok debt))
```

That would let them bypass the transfer of collaterals, mint some USDA + create a vault for free.

Assuming the reserves are being filled by honest users, an attacker could also create a vault, get the amount of STX not being stacked by the reserve, artificially adding a collateral with this value by passing a malicious reserve, and then withdraw by passing the official reserve contract, which would end up draining the reserve.

Recommendation

Ensure that the reserve can be trusted by maintaining a whitelist of contracts. Methods impacted: - `arkadiko-freddie-v1-1::collateralize-and-mint` - `arkadiko-freddie-v1-1::deposit` - `arkadiko-freddie-v1-1::withdraw` - `arkadiko-freddie-v1-1::mint` - `arkadiko-freddie-v1-1::burn` - `arkadiko-freddie-v1-1::close-vault` - `arkadiko-freddie-v1-1::redeem-auction-collateral` - `arkadiko-freddie-v1-1::withdraw-leftover-collateral`

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

0f-ARK-002: Increase protocol observability with events

Type	**note / suggestion**
Git commit	https://github.com/arkadiko-dao/arkadiko/commit/3a3d66012041da0cd386d16d7360aa863d38f25c

Context

Clarity events are probably under-documented, and yet they constitute a powerful mechanism for increasing observability of onchain protocols, while being cheap in terms of transaction cost.

Observability is a crucial requirement for engaging protocol participants. Events can be leveraged for building an ecosystem on top of the Arkadiko protocol. Some concrete examples: - Bot biding on vaults being liquidated - Bot relaying governance activity via email (proposals, etc) - Mobile wallet fully supporting the DIKO token could be relaying via push notification some of the events emitted by the protocol (under collateralized positions, new collateral types, protocol upgrades, etc). - Alarming system for the Arkadiko foundation, triggering alerts when detecting abnormal activity.

Arkadiko is involving a lot of contracts, so emitting topic based events could be an approach scaling nicely with the current / future complexity. ``` (print { topic: "governance", object: "proposal", action: "created", params: .... }) (print { topic: "vaults", object: "vault", action: "created", params: .... }) (print { topic: "tokens", object: "DIKO", action: "minted", params: .... })

etc ```

Recommendation

Emit CRUD events for the different objects being manipulated in the different contracts (proposals, vaults, tokens, swaps, etc).

0f-ARK-003: Ability to drain protocol rewards by manipulating LP tokens

Type	high severity
Git commit	https://github.com/arkadiko-dao/arkadiko/commit/3a3d66012041da0cd386d16d7360aa863d38f25c

Context

Arkadiko includes a third party open source AMM that let token holders becoming liquidity providers by depositing some tokens X + Y in the `aarkadiko-swap-v1-1` contract. When increasing / decreasing their positions in paired tokens pools, some LP tokens, representing their stake in the liquidity pool, are being minted / burnt accordingly.

In the revision being audited, malicious users can provide liquidity on one pair, and choose to receive a LP token from a different pool. When removing liquidity from the pool, malicious users can also choose to burn a LP token from a different pool.

When calling the methods `add-to-position` and `reduce-position` , 3 traits arguments are required: - token-x - token-y - lp-token-x-y In the revision being audited, the `lp-token-x-y` argument is being blindly trusted. As a consequence, a malicious X+Y token holder can be minting an infinite amount of any LP tokens, and use these artificial LP tokens for draining the pool of rewards generated by the protocol.

Demonstration

The entrypoint for increasing / decreasing a position in a pool, `arkadiko-swap-v1-1::add-to-position` and `arkadiko-swap-v1-1::reduce-position` have the following signature:

```
(define-public (add-to-position
    (token-x-trait <ft-trait>)
    (token-y-trait <ft-trait>)
    (swap-token-trait <swap-token>)
    (x uint)
    (y uint)
```

```
(define-public (reduce-position
    (token-x-trait <ft-trait>)
    (token-y-trait <ft-trait>)
    (swap-token-trait <swap-token>)
    (percent uint))
```

In this 2 functions, we're not ensuring that the argument `swap-token-trait` is correctly providing the address of the LP token corresponding to token-x and token-y. A malicious user could create a contract with an implementation for this trait of their own:

```
(define-public (burn
  (sender principal)
  (amount uint)
)
    (ok amount))
```

Then by successively calling the method `add-to-position` / `reduce-position` with the same amount of tokens and passing any valid LP token trait in `add-to-position` (XY / YZ / etc), and a malicious trait implementation in the `reduce-position` call, a user can end up minting an infinite amount of LP token XY, YZ, etc.

Recommendation

Ensure that the swap-token being passed is the address of the contract of the adequate LP token, using the component `swap-token` in `pairs-data-map` . Methods impacted: - `arkadiko-swap-v1-1::add-to-position` - `arkadiko-swap-v1-1::reduce-position`

Additionally this check should be performed in the methods: - `arkadiko-swap-v1-1::get-position` - `arkadiko-swap-v1-1::get-data`

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

0f-ARK-004: Users can deposit into any vault

Type	**low severity**
Git commit	https://github.com/arkadiko-dao/arkadiko/commit/3a3d66012041da0cd386d16d7360aa863d38f25c

Context

`arkadiko-freddie-v1-1` is the contract orchestrating vaults management. In the revision being audited, it appears that anyone can deposit more tokens in any vault, without any other side effects This issue appears to be a bug, but does not seems like something that could be directly exploited, beside an offchain bug in the transaction construction (intentional, or not), leading users to deposit collateral in a vault that is not the one they wanted to fund.

Recommendation

Ensure that a user owns the vault they wants to add funds to. Methods impacted: - `arkadiko-freddie-v1-1::deposit`

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

0f-ARK-005: Possible total debt versus maximum dept discrepanties

Type	**medium severity**
Git commit	https://github.com/arkadiko-dao/arkadiko/commit/3a3d66012041da0cd386d16d7360aa863d38f25c

Context

`arkadiko-freddie-v1-1` is the contract orchestrating vaults management. In the revision being audited, it appears that the total debt check is incorrect. This issue could lead to the `arkadiko-freddie-v1-1` contract becoming unusable if a STX "whale" ended up creating a vault. Arkadiko governance could, via vote, upgrade and unblock the protocol and increase the maximal debt, but this could create issues in the protocol's tokenomic.

Demonstration

The entrypoint for depositing collateral into a vault, `arkadiko-freddie-v1-1::deposit` has the following assertion:

```
    (asserts!
      (<
        (get total-debt collateral-type)
        (get maximum-debt collateral-type)
      )
      (err ERR-MAXIMUM-DEBT-REACHED)
    )
```

This check ensures that the current debt is not overflowing the maximal debt, instead of checking that the total debt post deposit is strictly less than maximum debt authorized.

Recommendation

Ensure that the assertion above is taking into account the amount of tokens being deposited / minted. Methods impacted: - `arkadiko-freddie-v1-1::deposit` - `arkadiko-freddie-v1-1::mint`

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

0f-ARK-006: LP tokens wSTX-DIKO and wSTX-USDA being staked by users are lost

Type	high severity
Git commit	https://github.com/arkadiko-dao/arkadiko/commit/3a3d66012041da0cd386d16d7360aa863d38f25c

Context

This issue is tracking a class of bugs that are sharing a root cause: the contracts `arkadiko-stake-pool-wstx-*-v1-1` are failing transferring tokens back to their owners when users try to `unstake` or `emergency-withdraw`.

Demonstration

The entrypoint for withdrawing stake LP tokens, `arkadiko-stake-pool-wstx-*-v1-1::unstake` and `arkadiko-stake-pool-wstx-*-v1-1::emergency-withdraw` are invoking a token transfer with:

```
      (try! (contract-call? .arkadiko-swap-token-wstx-usda transfer stake-amount (as-contract tx-sender) tx-sender none))
```

The `arkadiko-swap-token-wstx-*-v1-1::transfer` implementations are including the following assertion:

```
    (asserts! (is-eq tx-sender sender) (err ERR-NOT-AUTHORIZED))
```

Implying that any `unstake` / `emergency-withdraw` will end up with `(err ERR-NOT-AUTHORIZED)`, since the `tx-sender` in this case is not the pool contract.

Recommendation

Wrap the token transfer `(contract-call? ...)` in a `(as-contract ...)` construct. Methods impacted: - `arkadiko-stake-pool-wstx-usda-v1-1::unstake` - `arkadiko-stake-pool-wstx-usda-v1-1::emergency-withdraw` - `arkadiko-stake-pool-wstx-diko-v1-1::unstake` - `arkadiko-stake-pool-wstx-diko-v1-1::emergency-withdraw`

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

0f-ARK-007: Defective funds migration on contract upgrade

Type	**high severity**
Git commit	https://github.com/arkadiko-dao/arkadiko/commit/3a3d66012041da0cd386d16d7360aa863d38f25c

Context

This issue is tracking a class of bugs that are sharing a root cause: the contracts `arkadiko-auction-engine-v1-1`, `arkadiko-freddie-v1-1` and `arkadiko-sip10-reserve-v1-1` would fail migrating the funds being held on protocol upgrades.

Demonstration

The entrypoint `migrate-funds` for migrating tokens from current version, to the next version are invoking token transfers with:

```
      (contract-call? token transfer balance (as-contract tx-sender) (contract-of new-vault) none)
```

The `arkadiko-swap-token-wstx-*-v1-1::transfer` implementations are including the following assertion:

```
    (asserts! (is-eq tx-sender sender) (err ERR-NOT-AUTHORIZED))
```

Implying that any `migrate-funds` invokation would end up failing with a `(err ERR-NOT-AUTHORIZED)` error - the contextual `tx-sender` is the party broadcasting the transaction, instead of the expected contract holding tokens.

Recommendation

Wrap the token transfer `(contract-call? ...)` in a `(as-contract ...)` construct. Methods impacted: - `arkadiko-auction-engine-v1-1::migrate-funds` - `arkadiko-freddie-v1-1::migrate-funds` - `arkadiko-sip10-reserve-v1-1::migrate-funds`

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

0f-ARK-008: Successful bidders can tap into any reserve when redeeming a lot

Type	**high severity**
Git commit	https://github.com/arkadiko-dao/arkadiko/commit/3a3d66012041da0cd386d16d7360aa863d38f25c

Context

When a vault becomes under-collaterized, it ends up being eligible for liquidation. When entering the liquidated, an auction is being created and marked as opened: a number of lots, based on the amount of collateral tracked in the vault is being auctioned and can be sold to anyone submitting the best / right bid. When a lot is won, the owner can redeem them. The method for redeeming a won lot is missing a check, that would let users specifying any reserve, instead of the one corresponding to the collateral being auctioned.

Demonstration

The entrypoint for redeeming a lot, `arkadiko-auction-engine-v1-1::redeem-lot-collateral` have the following signature:

```
(define-public (redeem-lot-collateral
  (vault-manager <vault-manager-trait>)
  (ft <ft-trait>)
  (reserve <vault-trait>)
  (coll-type <collateral-types-trait>)
  (auction-id uint)
  (lot-index uint))
```

In this function, we're not ensuring that the argument `reserve` is correctly providing the address of the reserve corresponding to the token being used as a collateral.

Recommendation

Ensure that the `reserve` being passed is the address of the contract of the expected reserve. Method impacted: - `arkadiko-auction-engine-v1-1::redeem-lot-collateral`

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

0f-ARK-009: Improve use of optionals and responses

Type	**informational**
Git commit	https://github.com/arkadiko-dao/arkadiko/commit/3a3d66012041da0cd386d16d7360aa863d38f25c

Context

In multiple contracts, a `default-to` response is being constructed if an entry cannot be find when querying a map with `map-get?`. This synthetic entry is then being used, and lazily failing, on design. This approach does work, but has a higher cost than just unwrapping the optional and early exiting the control flow, which would also make the contract easier to read.

Example

Synthetic entry constructed with:
`(define-read-only (get-auction-by-id (id uint)) (default-to { id: u0, auction-type: "collateral", collateral-amount: u0, collater`

would be aborting a `redeem-lot-collateral` contract calls because of a `ERR-TOKEN-TYPE-MISMATCH` error:
`(asserts! (or (is-eq (unwrap-panic (contract-call? ft get-symbol)) token-string) (is-eq "STX" token-string) (is-eq "xSTX" token-s`

Instances

The pattern described above is being used in multiple place:

- (define-read-only (get-auction-by-id (id uint)))
- (define-read-only (get-last-bid (auction-id uint) (lot-index uint)))
- (define-read-only (get-winning-lots (owner principal)))
- (define-read-only (get-collateral-type-by-name (name (string-ascii 12))))
- (define-read-only (get-proposal-by-id (proposal-id uint)))
- (define-read-only (get-vault-by-id (id uint)))
- (define-read-only (get-stacking-payout (vault-id uint) (lot-index uint)))

Informational issues indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

0f-ARK-010: Document protocol capacities

Type	informational
Git commit	https://github.com/arkadiko-dao/arkadiko/commit/3a3d66012041da0cd386d16d7360aa863d38f25c

Context

The Arkadiko protocol, for very good reasons, is bounding its capacity (vaults, auctions, bids, etc). It could be worthwhile to document all these limits, and ensure cohesion / consistency across the protocol.

Instances (not exhaustive)

- Number of concurrent auctions limited to 1500
  `(var-set auction-ids (unwrap-panic (as-max-len? (append (var-get auction-ids) auction-id) u1500)))`
- Number of winning lots limited to 100
  `(map-set winning-lots { user: tx-sender } { ids: (unwrap-panic (as-max-len? (append (get ids lots) (tuple (auction-id auction`
- Number of vaults per user limited to 500 `(define-map vault-entries { user: principal } { ids: (list 500 uint) })`

Informational issues indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

0f-ARK-011: Use the most precise token scale available

Type	**low severity**
Git commit	https://github.com/arkadiko-dao/arkadiko/commit/3a3d66012041da0cd386d16d7360aa863d38f25c

Context

The Arkadiko contracts are dealing with the token decimal precision at the contract levels. Contracts are explicitly manipulating `cents` (USDA: 1 cent = 1000 tokens) and `STX` , instead of manipulating tokens (micro quantities) directly, and letting UI components dealing with display using the `get-decimals` method required in SIP10 tokens.

As a consequence, a token must have a value >= 0.01 usda for being a collateral. If a token falls under this limit, consequences are unclear.

This precision choice also impact code readability: it requires salting the codebase with 100/1000/100000 multipliers/dividers.

Recommendation

Use tokens / micro quantities everywhere, and let the UI resolve the decimal precision (balance of 100000 USDA tokens = 1 USDA).

0f-ARK-012: Redundancy with (as-contract) constructs

Type	**informational**
Git commit	https://github.com/arkadiko-dao/arkadiko/commit/3a3d66012041da0cd386d16d7360aa863d38f25c

Context

When a closure is passed to `(as-contract ...)`, all the subsequent calls to `tx-sender` refers to the contract calling this construct. In multiple contracts, the following pattern is constructed:

```
(as-contract (contract-call? <token> transfer u1 (as-contract tx-sender) recipient none)))
```

The inner `(as-contract ...)` is unnecessary.

Instances

```
(define-public (redeem-tokens (usda-amount uint) (diko-amount uint))
    ...
    (try! (as-contract (contract-call? .arkadiko-token transfer diko-amount (as-contract tx-sender) (contract-call? .arkadiko-dao get-
    (as-contract (contract-call? .usda-token transfer usda-amount (as-contract tx-sender) (contract-call? .arkadiko-dao get-payout-add
      )
    ...
    (as-contract (contract-call? .usda-token transfer usda-amount (as-contract tx-sender) (contract-call? .arkadiko-dao get-payout-add
```

```
(define-public (redeem-usda (usda-amount uint))
    ...
    (as-contract (contract-call? .usda-token transfer usda-amount (as-contract tx-sender) (contract-call? .arkadiko-dao get-payout-add
```

```
(define-public (request-stx-to-stack (name (string-ascii 256)) (requested-ustx uint))
    ...
    (as-contract
      (stx-transfer? requested-ustx (as-contract tx-sender) (unwrap-panic (contract-call? .arkadiko-dao get-qualified-name-by-name nam
```

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

0f-ARK-013: Unnecessary statements

Type	informational
Git commit	https://github.com/arkadiko-dao/arkadiko/commit/3a3d66012041da0cd386d16d7360aa863d38f25c

Unnescessary wrapping:

```
(lot-got-sold (if (>= usda (var-get lot-size))
        (ok u1)
        (ok u0)
...
(+ (unwrap-panic lot-got-sold) (get lots-sold auction)),
```

Unnecessary field merge:
`(merge auction { total-debt-burned: (get total-debt-raised auction), ends-at: (+ (get ends-at auction) blocks-per-day) })`

Unnessary `if`:

```
(define-private (remove-auction (auction-id uint))
  (if true
```

```
(define-private (subtract-stx-redeemable (token-amount uint))
  (if true
    (ok (var-set stx-redeemable (- (var-get stx-redeemable) token-amount)))
    (err u0)
  )
)
```

```
(define-private (resolve-stacking-amount (collateral-amount uint) (collateral-token (string-ascii 12)) (stack-pox bool))
  (if (and (is-eq collateral-token "STX") stack-pox)
    collateral-amount
    u0
  )
)
```

```
(if (and (> block-height redeem-period-start) (< block-height redeem-period-end) (not (is-eq redeem-period-start u0)))
      true
      false
    )
```

```
(if (map-set vault-entries { user: tx-sender } { ids: (filter remove-burned-vault entries) })
```

(`map-set` always returns `true`)

Informational issues indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

0f-ARK-014: Oracle partially updating new entries

Type	**low severity**
Git commit	https://github.com/arkadiko-dao/arkadiko/commit/3a3d66012041da0cd386d16d7360aa863d38f25c

Context

When the method `arkadiko-oracle-v1-1::update-price` is invoked, it should be keeping track of the block `last-block`. Instead, it's using a hard-coded `0` value.

```
(define-public (update-price (token (string-ascii 12)) (price uint))
    ...
    (map-set prices { token: token } { last-price-in-cents: price, last-block: u0 })
```

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

0f-ARK-015: Bouquet of various suggestions

Type	**informational**
Git commit	https://github.com/arkadiko-dao/arkadiko/commit/3a3d66012041da0cd386d16d7360aa863d38f25c

Context

This issue includes a list of various suggestions and informational notes.

Consider changing scope, from `public` to `private` for the method:

```
(define-public (close-auction
  (vault-manager <vault-manager-trait>)
  (coll-type <collateral-types-trait>)
  (auction-id uint)
```

Consider adding an assertion, asserting that the protocol has been interrupted in the method:

```
(define-public (migrate-funds (auction-engine <auction-engine-trait>) (token <ft-trait>))
```

The method `add-debt-to-collateral-type` returns its `debt` input, consider returning the total value instead - which could be more useful for the caller.

```
(define-public (add-debt-to-collateral-type (token (string-ascii 12)) (debt uint))
```

Consider changing the approach taken in this snippet:

```
(if (> (get total-debt collateral-type) debt)
    (map-set collateral-types { name: token } (merge collateral-type { total-debt: (- (get total-debt collateral-type) debt) }))
    (map-set collateral-types { name: token } (merge collateral-type { total-debt: u0 }))
)
```

The second branch in this `if` looks like a bug, consider raising an event / aborting.

Consider removing this methods if we're looking at dead code:

```
(define-public (add-collateral-type (token (string-ascii 12))
                                    (name (string-ascii 12))
                                    (url (string-ascii 256))
                                    (collateral-type (string-ascii 12))
                                    (token-address principal)
                                    (liquidation-ratio uint)
                                    (liquidation-penalty uint)
                                    (stability-fee uint)
                                    (stability-fee-decimals uint)
                                    (stability-fee-apy uint)
                                    (maximum-debt uint)
                                    (collateral-to-debt-ratio uint))
```

```
(define-public (return-stx (ustx-amount uint))
```

Consider removing minting and burning privileges to the guardian contract:

```
(map-set contracts
    { name: "diko-guardian" }
    {
      address: 'ST1PQHQKV0RJXZFY1DGX8MNSNYVE3VGZJSRTPGZGM,
      qualified-name: 'ST1PQHQKV0RJXZFY1DGX8MNSNYVE3VGZJSRTPGZGM.arkadiko-diko-guardian-v1-1
    }
  )

(map-set contracts-data
    { qualified-name: 'ST1PQHQKV0RJXZFY1DGX8MNSNYVE3VGZJSRTPGZGM.arkadiko-diko-guardian-v1-1 }
    {
      can-mint: true,
      can-burn: true
    }
  )
```

Consider changing the action from "burn" to "close" in the event:

```
(define-public (close-vault
  (vault-id uint)
  (reserve <vault-trait>)
  (ft <ft-trait>)
  (coll-type <collateral-types-trait>)
)
    ...
    (print { type: "vault", action: "burn", data: updated-vault })
```

Consider removing the unused `reserve` argument from the method:

```
(define-private (burn-partial-debt
  (vault-id uint)
  (debt uint)
  (reserve <vault-trait>)
  (ft <ft-trait>)
  (coll-type <collateral-types-trait>)
```

Consider removing one of the redundant check in the methods:

```
(define-public (notify-risky-vault
  (vault-manager <vault-manager-trait>)
  (auction-engine <auction-engine-trait>)
  (vault-id uint)
  (coll-type <collateral-types-trait>)
  (oracle <oracle-trait>)
)
    ...
    (asserts! (is-eq (unwrap-panic (contract-call? .arkadiko-dao get-emergency-shutdown-activated)) false) (err ERR-EMERGENCY-SHUTDOWN
```

```
(define-public (liquidate
  (vault-id uint)
  (coll-type <collateral-types-trait>)
)
    ...
    (asserts!
      (and
        (is-eq (unwrap-panic (contract-call? .arkadiko-dao get-emergency-shutdown-activated)) false)
        (is-eq (var-get freddie-shutdown-activated) false)
      )
      (err ERR-EMERGENCY-SHUTDOWN-ACTIVATED)
    )
```

Consider removing the unused `debt` argument from the method:

```
(define-public (collateralize-and-mint
  (token <ft-trait>)
  (token-string (string-ascii 12))
  (ucollateral-amount uint)
  (debt uint)
  (sender principal)
  (stacker-name (string-ascii 256))
  (stack-pox bool)
```

Note that this function is also not in charge of actual minting. Consider revisiting the naming.

Consider cleaning debug events leftovers:

```
(define-public (initiate-stacking (pox-addr (tuple (version (buff 1)) (hashbytes (buff 20))))
                                  (start-burn-ht uint)
                                  (lock-period uint))
    ...
    (print result) ;;;; Leftover?
```

Consider publishing a JSON at the URL

```
(define-read-only (get-token-uri)
  (ok (some u"https://arkadiko.finance/tokens/diko-usda-token.json"))
)
```

Consider automatically updating the DIKO token owner when the dao-owner is updated

```
(define-public (set-dao-owner (address principal))
```

Consider updating the comments referring to SRC20 Standard (instead of SIP10)

```
;; Defines an STX derivative according to the SRC20 Standard
```

Consider removing hard-coded returned value in the method:

```
(define-public (subtract-tokens-to-stack (name (string-ascii 256)) (token-amount uint))
    ...
    (ok u200)
```

Consider gating Testnet constants

```
(define-public (initiate-stacking (pox-addr (tuple (version (buff 1)) (hashbytes (buff 20))))
                                  (start-burn-ht uint)
                                  (lock-period uint))
    ...
    (match (as-contract (contract-call? 'ST000000000000000000002AMW42H.pox can-stack-stx pox-addr tokens-to-stack start-burn-ht lock-p
```

```
(begin
  ;; TODO: do not do this on testnet or mainnet
  (try! (ft-mint? diko u890000000000 'ST1PQHQKV0RJXZFY1DGX8MNSNYVE3VGZJSRTPGZGM))
  (try! (ft-mint? diko u150000000000 'ST1SJ3DTE5DN7X54YDH5D64R3BCB6A2AG2ZQ8YPD5))
  (try! (ft-mint? diko u150000000000 'ST2CY5V39NHDPWSXMW9QDT3HC3GD6Q6XX4CFRK9AG))
  (try! (ft-mint? diko u1000000000000 'STB2BWB0K5XZGS3FXVTG3TKS46CQVV66NAK3YVN8))
  (try! (ft-mint? diko u1000000000000 'ST1QV6WVNED49CR34E58CRGA0V58X281FAS1TFBWF))
)
```

```
;; Initialize the contract
(begin
  ;; TODO: do not do this on testnet or mainnet
  (try! (ft-mint? usda u10 'ST2JHG361ZXG51QTKY2NQCVBPPRRE2KZB1HR05NNC))
  (try! (ft-mint? usda u1000000000000 'ST1PQHQKV0RJXZFY1DGX8MNSNYVE3VGZJSRTPGZGM)) ;; 1 million USDA
  (try! (ft-mint? usda u1000000000000 'ST1QV6WVNED49CR34E58CRGA0V58X281FAS1TFBWF)) ;; 1 million USDA
  (try! (ft-mint? usda u1000000000000 'ST2CY5V39NHDPWSXMW9QDT3HC3GD6Q6XX4CFRK9AG)) ;; 1 million USDA
  (try! (ft-mint? usda u1000000000000 'ST1SJ3DTE5DN7X54YDH5D64R3BCB6A2AG2ZQ8YPD5)) ;; 1 million USDA
)
```

Informational issues indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Appendix A

Remediations

Issue	Link
0F-ARK-001	https://github.com/arkadiko-dao/arkadiko/pull/192
0F-ARK-002	https://github.com/arkadiko-dao/arkadiko/pull/208
0F-ARK-003	https://github.com/arkadiko-dao/arkadiko/pull/221
0F-ARK-004	https://github.com/arkadiko-dao/arkadiko/pull/243
0F-ARK-005	https://github.com/arkadiko-dao/arkadiko/pull/214
0F-ARK-006	https://github.com/arkadiko-dao/arkadiko/pull/215
0F-ARK-007	https://github.com/arkadiko-dao/arkadiko/pull/226
0F-ARK-008	https://github.com/arkadiko-dao/arkadiko/pull/205
0F-ARK-009	https://github.com/arkadiko-dao/arkadiko/pull/246
0F-ARK-010	https://github.com/arkadiko-dao/arkadiko/pull/239
0F-ARK-011	https://github.com/arkadiko-dao/arkadiko/pull/227
0F-ARK-012	https://github.com/arkadiko-dao/arkadiko/pull/245
0F-ARK-013	https://github.com/arkadiko-dao/arkadiko/pull/244
0F-ARK-014	https://github.com/arkadiko-dao/arkadiko/pull/228
0F-ARK-015	https://github.com/arkadiko-dao/arkadiko/pull/246