BSD Money Clarity Smart Contract Audit Report

author: Nolan(X:@ma1fan)

date: Nov 25, 2024

website:https://www.bsd.money/

Table of Contents

    - Medium
    - 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	**BSD Money Clarity Smart Contract Audit Report**
Repository	https://github.com/bsdmoney/bsd-contracts
Commit	**e87d3608027a5207cdccbcc38b391d105b85b5b5**
Audit Timeline	**Oct 18 - Nov 20 th**
Methods	**Manual Review, Security Testing**

Issues Found

	Count
Critical Risk	3
High Risk	8
Medium Risk	1
Low Risk	2
Informational	0
Total Issues	**14**

Summary of Findings

Severity	Description	Status
High	1.Function `set-redeem-parameters` should make sure `new-min-redeem-fee` smaller than `new-max-redeem-fee`	Fixed
High	2.The function `set-protocol` does not check whether the new-state is valid or not ,	Fixed
High	3.Function `set-borrow-parameters` should make sure `new-min-borrow-fee` smaller than `new-max-borrow-fee`	Fixed
Low	4.Function `sbtc-collateral-pre-fee` `redeem-fee` can be controlled and they can be equal,so redeem-to-user may be 0.	Fixed
High	5. `new-global-collateral-ratio-threshold` should make sure bigger than 100%	Fixed
High	6.Precision Loss exists in function `update-base-rate`	Fixed
High	7. In function `attempt-liquidate-vault` we should not assign 1 as argument when vault-increased-bsd is 0	Fixed
Critical	8.In this function, we should make sure `new-vault-recovery-ratio-threshold` is bigger than `new-vault-collateral-ratio-threshold` otherwise it will lead to vault funds lost	Fixed
Critical	9.NO auth check when `vault-liquidated` is true in the `withdraw-collateral-wrapper`	Fixed
Critical	10.In the function `remove-liquidity` , should check the amount is not bigger than `liquidity-staked`	Fixed
High	11.It should not simply assign 1 as the argument (if protocol-debt is 0)when calculating `vault-share` ,	Fixed
Low	12.In the function `withdraw-collateral-wrapper` contains no use code, `new-total-collateral-in-sbtc` seems not used anywhere. Should delete if the code is useless	Fixed
High	13. In the function `get-protocol-data` , if `total-debt-bsd` is 0, will assign denominator 1	Fixed
Medium	14.In the function add-liquidity. If the amount is 0, it can still execute success,and may lead to creating may useless `stability-pool-providers` and waste gas.	Fixed

Findings

1. High ,Function `set-redeem-parameters` should make sure `new-min-redeem-fee` smaller than `new-max-redeem-fee`

```
(define-public (set-redeem-parameters (new-min-redeem-fee uint) (new-max-
redeem-fee uint) (new-alpha uint) (new-min-redeem-amount uint))
    (begin
        (try! (contract-call? .controller-v1-0 is-admin tx-sender))
        (var-set min-redeem-fee new-min-redeem-fee)
        (var-set max-redeem-fee new-max-redeem-fee)
        (var-set alpha new-alpha)
        (var-set min-redeem-amount new-min-redeem-amount)
        (ok true)
    )
)
```

Another Recommend: set `min-redeem-fee` `max-redeem-fee` value can be separate function. So if we only want to update one of them. No need to set another one again which controls the risk minimizes.

Status: Fixed

2.High , the function `set-protocol` does not check whether the new-state is valid or not ,

it should check the new-state is only can be 0<= new-state <=2,if someone input a value

```
;; pause-protocol
(define-public (set-protocol-state (new-state uint))
    (begin
         (try! (contract-call? .controller-v1-0 is-admin tx-sender))
        (var-set protocol-state new-state)
        (ok true)
    )
)

```

e.g. If we set the value is 3, is-paused is-maintenance of course is false

```

;; get-protocol-attributes
(define-read-only (get-protocol-attributes)
        (ok {
            ;; active protocol data
            total-bsd-loans: (get debt-bsd (var-get aggregate-debt-and-collateral)),
            total-sbtc-collateral: (get collateral-sbtc (var-get aggregate-debt-and-collateral)),
            active-vaults: (len (var-get active-vaults)),
            created-vaults: (var-get created-vaults),
            is-paused: (is-eq PROTOCOL_STATE_PAUSED (var-get protocol-state)),
            is-maintenance: (is-eq PROTOCOL_STATE_MAINTENANCE (var-get protocol-state)),
            base-rate: (var-get base-rate),
            last-redeem-height: (var-get last-redeem-height),

```

In the funtion `add-liquidity-wrapper` , the check will passed. In an emergency situation will lead to a serious problem.

```
;; add-liquidity
(define-public (add-liquidity-wrapper (amount uint) (bsd <bsd-trait>) (registry <registry-trait>))
    (let
        (
            (valid-registry (try! (contract-call? .controller-v1-0 check-approved-contract "registry" (contract-of registry))))
            (valid-bsd (try! (contract-call? .controller-v1-0 check-approved-contract "bsd" (contract-of bsd))))
            (current-provider (unwrap! (contract-call? registry get-stability-pool-provider tx-sender) ERR_STABILITY_PROVIDER_NOT_FOUND))
            (provider-balance (if (is-eq current-provider none) u0 (unwrap! (get liquidity-staked current-provider) ERR_STABILITY_PROVIDER_NOT_FOUND)))
            (protocol-attributes (unwrap-panic (contract-call? registry get-protocol-attributes)))
            (min-stability-provider-balance (get min-stability-provider-balance protocol-attributes))
            (is-paused (get is-paused protocol-attributes))
            (is-maintenance (get is-maintenance protocol-attributes))
        )

        ;; check paused
        (asserts! (not is-paused) ERR_PROTOCOL_STATE)

        ;; check maintenance
        (asserts! (not is-maintenance) ERR_PROTOCOL_STATE)

        (asserts! (>= (+ provider-balance amount) min-stability-provider-balance) ERR_MIN_BALANCE)

        ;; transfer bsd to the stability pool
        (try! (contract-call? bsd protocol-transfer amount tx-sender (as-contract tx-sender)))
        ;; call registry to complete
        (try! (contract-call? registry add-liquidity amount tx-sender))
        (ok
            (unwrap-panic (contract-call? registry get-stability-pool-provider tx-sender))
        )
    )
)
```

Status: Fixed

3. High , Function `set-borrow-parameters` should make sure `new-min-borrow-fee` smaller than `new-max-borrow-fee`

```
(define-public (set-borrow-parameters (new-min-borrow-fee uint) (new-max-borrow-fee uint) (new-loan-minimum uint))
    (begin
        (try! (contract-call? .controller-v1-0 is-admin tx-sender))
        (var-set min-borrow-fee new-min-borrow-fee)
        (var-set max-borrow-fee new-max-borrow-fee)
        (var-set vault-loan-minimum new-loan-minimum)
        (ok true)
    )
)
```

Another Recommend: set `new-min-borrow-fee` `new-max-borrow-fee` value can be separate function. So if we only want to update one of them. No need to set another one again which controls the risk minimizes.

Note: The same problem in function `set-vault-parameters`

Status: Fixed

4.Low, `sbtc-collateral-pre-fee` `redeem-fee` can be controlled and they can be equal,so redeem-to-user may be 0.

```
;; calculate-redeem-info
(define-private (calculate-redeem-info (redeem-bsd uint) (sbtc-price-in-bsd uint) (registry <registry-trait>))
    (let (
            (valid-registry (try! (contract-call? .controller-v1-0 check-approved-contract "registry" (contract-of registry))))
            (elapsed-blocks (contract-call? registry get-height-since-last-redeem))
            (calc-base-rate (try! (contract-call? registry calculate-redeem-fee-rate redeem-bsd)))
            (sbtc-collateral-pre-fee (contract-call? .math-v1-0 div-to-fixed-precision redeem-bsd PRECISION sbtc-price-in-bsd))
            (redeem-fee (contract-call? .math-v1-0 mul-perc calc-base-rate u8 sbtc-collateral-pre-fee))
            (redeem-to-user (- sbtc-collateral-pre-fee redeem-fee))
        )
        (ok {
                redeem-fee: redeem-fee,
                redeem-to-user: redeem-to-user,
                base-rate: calc-base-rate
            })
    )
)

```

And this function was called in `redeem-wrapper` ,so in here if redeem-to-user is 0, we should return error.

```
        ;; Call 'protocol-burn-bsd' from the token contract to burn the bsd
        (try! (contract-call? bsd protocol-burn tx-sender bsd-amount))

        ;; Transfer the sbtc to the user
        (try! (contract-call? vault protocol-transfer tx-sender redeem-to-user sbtc registry))
```

```
```

Status: Fixed

5. High, `new-global-collateral-ratio-threshold` should make sure bigger than 100%

```
(define-public (set-global-parameters (new-global-collateral-ratio-threshold uint) (new-global-collateral-cap uint) (new-protocol-fee-destination principal) (new-min-stability-provider-balance uint) (new-epoch-genesis uint))
    (begin
        (try! (contract-call? .controller-v1-0 is-admin tx-sender))
        (var-set global-collateral-ratio-threshold new-global-collateral-ratio-threshold)
        (var-set global-collateral-cap new-global-collateral-cap)
        (var-set protocol-fee-destination new-protocol-fee-destination)
        (var-set min-stability-provider-balance new-min-stability-provider-balance)
        (var-set epoch-genesis new-epoch-genesis)
        (ok true)
    )
)
```

Status: Fixed

6. High, Precision Loss exist in function `update-base-rate`

total-bsd-loans may be bigger than (* redeem-amount u100), so this may lead to redeem-over-global is 0,

This can lead to incorrect calculations.

```
;; update-base-rate
;; description: Helper function to update the base rate before/after a redemption
;; increase b(t) = b(t-1) + alpha * (m/n)
;; inputs: redeem-amount/uint - the amount of bsd being redeemed
(define-private (update-base-rate (current-base-rate uint) (redeem-amount uint) (registry <registry-trait>))
    (let
        (
            (protocol-attributes (unwrap-panic (contract-call? registry get-protocol-attributes)))
            (base-rate-constants (unwrap-panic (contract-call? registry get-base-rate-constants)))
            (redeem-over-global (/ (* redeem-amount u100) (get total-bsd-loans protocol-attributes)))
            (increase-amount (* (get alpha base-rate-constants) redeem-over-global))
            (base-rate-increased (+ current-base-rate increase-amount))
            (height-since-last-update (contract-call? registry get-height-since-last-redeem))
        )
            (ok base-rate-increased)
    )
)

```

Status: Fixed

7. High, In function `attempt-liquidate-vault ` we should not assign 1 as argument when vault-increased-bsd is 0

Code: https://github.com/bsdmoney/bsd-contracts/blob/690ca38b5ccbb7c1efe8b8bc959509b624ade044/clarity/contracts/protocol/v1/vault-v1-0.clar#L709

if assign 1 as b-fixed argument which means the return function mul-to-fixed-precision value is equal to `vault-collateral-in-usd`,

```
(define-read-only (mul-to-fixed-precision (a uint) (decimals-a uint) (b-fixed uint))
  (if (> decimals-a fixed-precision)
    (mul (/ a (pow u10 (- decimals-a fixed-precision))) b-fixed)
    (mul (* a (pow u10 (- fixed-precision decimals-a))) b-fixed)
  )
)
```

so it will make the `vault-collateral-ratio` very big

Status: Fixed

8. Critical, In this function, we should make sure `new-vault-recovery-ratio-threshold` is bigger than `new-vault-collateral-ratio-threshold` otherwise it will lead to vault funds lost

```
(define-public (set-vault-parameters (new-interest-minimum uint) (new-interest-maximum uint) (new-vault-collateral-ratio-threshold uint) (new-vault-recovery-ratio-threshold uint))
    (begin
        (try! (contract-call? .controller-v1-0 is-admin tx-sender))
        (var-set vault-collateral-ratio-threshold new-vault-collateral-ratio-threshold)
        (var-set vault-recovery-ratio-threshold new-vault-recovery-ratio-threshold)
        (var-set vault-interest-minimum new-interest-minimum)
        (var-set vault-interest-maximum new-interest-maximum)
        (ok true)
    )
)
```

Status: Fixed

9. Critical, NO auth check when `vault-liquidated` is true in the `withdraw-collateral-wrapper`

In the function, only check auth when `vault-liquidated` is false . If another contract calls the function `vault-liquidated` is true , because in this situation . There is no check if the tx-sender is equal to special vault-id borrower, Attacker can get arbitrary vault-data, image if another trusts the return data and does other finance calculate, which will lead to funds lost!

```
;; withdraw-collateral
(define-public (withdraw-collateral-wrapper (vault-id uint) (collateral-sbtc uint) (bsd <bsd-trait>) (sbtc <sbtc-trait>) (oracle <oracle-trait>) (registry
<registry-trait>) (stability <stability-trait>))
    (let
        (
            (valid-stability (try! (contract-call? .controller-v1-0 check-approved-contract "stability" (contract-of stability))))
            (valid-registry (try! (contract-call? .controller-v1-0 check-approved-contract "registry" (contract-of registry))))
            (valid-oracle (try! (contract-call? .controller-v1-0 check-approved-contract "oracle" (contract-of oracle))))
            (valid-bsd (try! (contract-call? .controller-v1-0 check-approved-contract "bsd" (contract-of bsd))))
            (valid-sbtc (try! (contract-call? .controller-v1-0 check-approved-contract "sbtc" (contract-of sbtc))))
            (sbtc-price (try! (contract-call? oracle get-price BTC_TOKEN_KEY)))
            (protocol-data (unwrap! (contract-call? registry get-protocol-data sbtc-price) ERR_NO_PROTOCOL_DATA))
            (recovery-mode (get recovery-mode protocol-data))
            (vault-data (unwrap! (contract-call? registry get-vault vault-id) ERR_VAULT_NOT_FOUND))
            (new-vault-collateral (- (unwrap-panic (get collateral-sbtc vault-data)) collateral-sbtc))
            (new-vault-collateral-in-usd (contract-call? .math-v1-0 mul-to-fixed-precision new-vault-collateral PRECISION sbtc-price))
            (new-total-collateral-in-sbtc (- (get total-sbtc-collateral protocol-data) collateral-sbtc))
            (loan-bsd (unwrap-panic (get borrowed-bsd vault-data)))
            (current-ratio (contract-call? .math-v1-0 div-to-fixed-precision (if (is-eq u0 new-vault-collateral-in-usd) u200 new-vault-collateral-in-usd) PRECISION (if (is-eq loan-bsd u0) u1 loan-bsd)))
            (recovery-threshold (get recovery-threshold protocol-data))
            (vault-liquidated (try! (liquidate-or-accrue vault-id sbtc-price bsd sbtc registry stability)))
            (is-paused (get is-paused protocol-data))
            (vault-threshold (if recovery-mode recovery-threshold (get vault-threshold protocol-data)))
            (vault-loan-minimum (get vault-loan-minimum protocol-data))
            (vault-collateral-minimum-usd (contract-call? .math-v1-0 mul-perc vault-loan-minimum PRECISION vault-threshold))
        )

        ;; check paused
        (asserts! (not is-paused) ERR_PROTOCOL_STATE)

        (print {
            current-sbtc-price: sbtc-price,
            new-vault-collateral: new-vault-collateral,
            new-vault-collateral-in-usd: new-vault-collateral-in-usd,
            new-total-collateral-in-sbtc: new-total-collateral-in-sbtc,
            recovery-mode: recovery-mode,
            recovery-threshold: recovery-threshold,
            vault-threshold: vault-threshold,
            current-ratio: current-ratio,
        })

        ;; return if vault has been liquidated
        (if (is-eq vault-liquidated true)

            ;; vault has been liquidated
            (ok
                {
                    vault-id: vault-id,
                    liquidated: true,
                    information: (unwrap-panic (contract-call? registry get-vault vault-id))
                }
            )

            ;; vault has not been liquidated - proceed to withdraw collateral
            (begin

                ;; check that tx-sender is the owner of the vault
                (asserts! (is-eq tx-sender (unwrap-panic (get borrower vault-data))) ERR_NOT_AUTH)

```

Should check tx-sender is borroer even though vault-liquidated is true

Status: Fixed

10. Critical, In the function `remove-liquidity` , should check the amount is not bigger than `liquidity-staked`

```
;; remove-liquidity
(define-public (remove-liquidity (amount uint) (provider principal))
    (let
        (
            (current-provider (unwrap-panic (map-get? stability-pool-providers provider)))
            (current-stability-pool (var-get stability-pool))
            (decreased-aggregate (- (get aggregate current-stability-pool) amount))
        )
        ;; Check that caller is protocol-caller
        (try! (contract-call? .controller-v1-0 is-protocol-caller contract-caller))
        ;; ;; Update stability pool aggregate
        (var-set stability-pool (merge
            current-stability-pool
            { aggregate: decreased-aggregate }
        ))
        ;; ;; check if all liquidity is removed
        (ok (if (is-eq amount (get liquidity-staked current-provider))
            ;; all liquidity & rewards are removed, must delete map & update list
            (begin
                ;; Remove provider map entry
                (map-delete stability-pool-providers provider)
                ;; Update stability pool aggregate & remove provider from active list
                (var-set stability-pool {
                    aggregate: decreased-aggregate,
                    active: (get new-list (try! (fold remove-principal-from-list (get active current-stability-pool) (ok {found: false, compare-principal: provider, new-list: (list )})))),
                })
            )
            ;; liquidity remains, update map
            (map-set stability-pool-providers provider (merge
                current-provider
                { liquidity-staked: (- (get liquidity-staked current-provider) amount) }
            ))
        ))
    )
)

```

Status: Fixed

11. High, It should not simply assign 1 as the argument (if protocol-debt is 0)when calculating `vault-share` ,

If I'm not mistaken, we should make sure that `protocol-debt` is greater than or equal to `current-vault-bsd` ?

```
;; redistribute-remaining-vault-debt-and-collateral
(define-private (redistribute-remaining-vault-debt-and-collateral (vault-id uint) (helper-tuple {redistributed-bsd-amt: uint, redistributed-collateral-amt: uint, bsd-aggregate: uint, liquidated-vault: uint}))
    (let
        (
            (current-vault (unwrap-panic (map-get? vault vault-id)))
            (current-vault-bsd (get borrowed-bsd current-vault))
            (protocol-debt (get bsd-aggregate helper-tuple))
            ;; avoid div/0
            (vault-share (contract-call? .math-v1-0 div-to-fixed-precision current-vault-bsd PRECISION (if (is-eq u0 protocol-debt) u1 protocol-debt)))
            (bsd-distribution (contract-call? .math-v1-0 mul-perc vault-share PRECISION (get redistributed-bsd-amt helper-tuple)))
            (current-vault-collateral (get collateral-sbtc current-vault))
            (sbtc-distribution (contract-call? .math-v1-0 mul-perc vault-share PRECISION (get redistributed-collateral-amt helper-tuple)))

        )


```

Status: Fixed

12. Low, In the function `withdraw-collateral-wrapper` contain no use code

`new-total-collateral-in-sbtc` seems not used anywhere. Should delete if the code is useless

```
            (new-total-collateral-in-sbtc (- (get total-sbtc-collateral protocol-data) collateral-sbtc))
```

Status: Fixed

13.High, In the function `get-protocol-data`, if `total-debt-bsd` is 0, will assign denominator 1, so this will Lead global-ratio very huge, so this will be very easy control the recovery-mode is false in the function

```
;; get-protocol-data
(define-read-only (get-protocol-data (sbtc-price uint))
    (let
        (
            (aggregate-amounts (var-get aggregate-debt-and-collateral))
            (total-debt-bsd (get debt-bsd aggregate-amounts))
            (total-collateral-in-sbtc (get collateral-sbtc aggregate-amounts))
            (total-collateral-in-bsd (contract-call? .math-v1-0 mul-perc total-collateral-in-sbtc PRECISION sbtc-price))
            (global-threshold (var-get global-collateral-ratio-threshold))
            (denominator (if (is-eq u0 total-debt-bsd) u1 total-debt-bsd))
            (global-ratio (contract-call? .math-v1-0 div-to-fixed-precision total-collateral-in-bsd PRECISION denominator))
            (recovery-mode (< global-ratio global-threshold))
        )
        (ok
            (merge
                {
                    current-oracle-price-sbtc: sbtc-price,
                    global-ratio: global-ratio,
                    recovery-mode: recovery-mode,
                    total-collateral-in-bsd: total-collateral-in-bsd,
                }
                (unwrap-panic (get-protocol-attributes))
            )
        )
    )
)
```

And in the caller function `withdraw-collateral-wrapper` , in line 24 (code below) , the vault-threshold will always be equal to `(get vault-threshold protocol-data)` (line 22), This can lead to incorrect collateral ratios and allow attacker withdraw `collateral-sbtc` which is unexpected.

```
1 ;; withdraw-collateral
2 (define-public (withdraw-collateral-wrapper (vault-id uint) (collateral-sbtc uint) (bsd <bsd-trait>) (sbtc <sbtc-trait>) (oracle <oracle-trait>) (registry <registry-trait>) (stability <stability-trait>))
3     (let
4         (
5             (valid-stability (try! (contract-call? .controller-v1-0 check-approved-contract "stability" (contract-of stability))))
```

```
            (valid-registry (try! (contract-call? .controller-v1-0 check-approved-contract "registry" (contract-of registry))))
            (valid-oracle (try! (contract-call? .controller-v1-0 check-approved-contract "oracle" (contract-of oracle))))
            (valid-bsd (try! (contract-call? .controller-v1-0 check-approved-contract "bsd" (contract-of bsd))))
            (valid-sbtc (try! (contract-call? .controller-v1-0 check-approved-contract "sbtc" (contract-of sbtc))))
            (sbtc-price (try! (contract-call? oracle get-price BTC_TOKEN_KEY burn-block-height registry)))
            (accrued-interest (try! (accrue-vault vault-id bsd registry)))
            (protocol-data (unwrap! (contract-call? registry get-protocol-data sbtc-price) ERR_NO_PROTOCOL_DATA))
            (recovery-mode (get recovery-mode protocol-data))
            (vault-data (unwrap! (contract-call? registry get-vault vault-id) ERR_VAULT_NOT_FOUND))
            (current-vault-collateral (unwrap-panic (get collateral-sbtc vault-data)))
            (valid-removal-amount (asserts! (and (> collateral-sbtc u0) ( >= current-vault-collateral collateral-sbtc)) ERR_INVALID_INPUT))
            (new-vault-collateral (- current-vault-collateral collateral-sbtc))
            (new-vault-collateral-in-usd (contract-call? .math-v1-0 mul-to-fixed-precision new-vault-collateral PRECISION sbtc-price))
            (new-total-collateral-in-sbtc (- (get total-sbtc-collateral protocol-data) collateral-sbtc))
            (current-debt-bsd (unwrap-panic (get borrowed-bsd vault-data)))
            (current-ratio (contract-call? .math-v1-0 div-to-fixed-precision new-vault-collateral-in-usd PRECISION u1))
            (recovery-threshold (get recovery-threshold protocol-data))
            (is-paused (get is-paused protocol-data))
            (vault-threshold (if recovery-mode recovery-threshold (get vault-threshold protocol-data)))
            (vault-loan-minimum (get vault-loan-minimum protocol-data))
            (vault-collateral-minimum-usd (contract-call? .math-v1-0 mul-perc vault-loan-minimum PRECISION vault-threshold))
        )
```

Status: Fixed

14.Medium, In the function add-liquidity. If the amount is 0, it can still execute success,and may lead to creating may useless `stability-pool-providers` and waste gas.

it should check the amount is 0 or not, if the amout is 0, should return error.

```
;; add-liquidity
(define-public (add-liquidity (amount uint) (provider principal))
    (let
        (
            (current-provider (map-get? stability-pool-providers provider))
            (current-stability-pool (var-get stability-pool))
            (increased-aggregate (+ (get aggregate current-stability-pool) amount))
        )
        ;; Check that caller is protocol-caller
        (try! (contract-call? .controller-v1-0 is-protocol-caller contract-caller))
        ;; Different paths for new provider & existing provider
        (ok (match current-provider
            existing-provider
            (begin
                ;; Update existing provider map entry
                (map-set stability-pool-providers provider (merge
                    existing-provider
                    { liquidity-staked: (+ (get liquidity-staked existing-provider) amount) }
                ))
                ;; Update stability pool aggregate
                (var-set stability-pool {
                    aggregate: increased-aggregate,
                    active: (get active current-stability-pool),
                })
            )
            (begin
                ;; Create new provider map entry
                (map-set stability-pool-providers tx-sender {
                    liquidity-staked: amount,
                    rewards-to-claim: u0,
                    last-claimed: burn-block-height,
                })
                ;; Update stability pool aggregate & add provider to active list
                (var-set stability-pool {
                    aggregate: increased-aggregate,
                    active: (unwrap! (as-max-len? (append (get active current-stability-pool) tx-sender) u1000) ERR_LIST_OVERFLOW),
                })
                )
            ))
    )
)
```

Status: Fixed