ClarityAlliance

STACKINGDAO BTC YIELDING STX LST SECURITY REVIEW

Conducted by:
KRISTIAN APOSTOLOV, ALIN BARBATEI (ABA)

JANUARY 9TH, 2025

CONTENTS

1. About Clarity Alliance

Clarity Alliance is a team of expert whitehat hackers specialising in securing protocols on Stacks.

They have disclosed vulnerabilities that have saved millions in live TVL and conducted thorough reviews for some of the largest projects across the Stacks ecosystem.

Learn more about Clarity Alliance at clarityalliance.org.

ClarityAlliance
Security Review

StackingDAO BTC Yielding STX LST

CONTENTS

ClarityAlliance
Security Review

StackingDAO BTC
Yielding STX LST

2. Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Clarity Alliance to perform a security assessment.

This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Clarity Alliance’s position is that each company and individual are responsible for their own due diligence and continuous security. Clarity Alliance’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree
to analyze.

The assessment services provided by Clarity Alliance are subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis.

Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third parties. Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Clarity Alliance does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

CONTENTS

3. Introduction

A time-boxed security review of the StackingDAO BTC Yielding STX LST, where Clarity Alliance reviewed the scope and providing insight on improving the protocol.

4. About StackingDAO

Stacking DAO is a Liquid Stacking protocol on Stacks that makes Stacking easily accessible to anyone and unlocks liquidity for Stacked STX through stSTX, which can then be used across DeFi.

How does it work?

- Anyone can participate in Stacking by depositing STX into the protocol.
- Users will then receive stSTX, a liquid representation of stacked STX that accrues in value as Stacking rewards are collected.
- Finally, stSTX can also be used across DeFi to earn additional yield.

CONTENTS

5. Risk Classification

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

5.1 Impact

- High - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- Medium - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- Low - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

5.2 Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

5.3 Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

CONTENTS

6. Security Assessment Summary

Review Commit Hash:
64853f3bbf709b978170ff1630d890ce65f177a3

ClarityAlliance
Security Review

StackingDAO BTC Yielding STX LST

CONTENTS

7. Executive Summary

Over the course of the security review, Kristian Apostolov, ABA engaged with - to review StackingDAO. In this period of time a total of **8** issues were uncovered.

Protocol Summary

Protocol Name	StackingDAO
Date	January 9th, 2025

Findings Count

Severity	Amount
High	1
Medium	4
Low	1
QA	2
Total Findings	**8**

CONTENTS

Summary of Findings

ID	Title	Severity	Status
[H-01]	Commission Contract Validation Leads to Locked Rewards	High	Resolved
[M-01]	stSTX Price Vulnerable to Manipulation with Flashloan Access to Entire Supply	Medium	Resolved
[M-02]	Semi-Trusted Position-Managing Contracts Lack Balance Validation	Medium	Resolved
[M-03]	Positions That Are Subsequently Un-Whitelisted Can Replay Reward Claims	Medium	Resolved
[M-04]	Holders in Non-Whitelisted Positions Could Continue Claiming sBTC Rewards	Medium	Resolved
[L-01]	Position Input Not Validated as position-trait	Low	Resolved
[QA-01]	Incorrect Parameter Naming	QA	Resolved
[QA-02]	Unnecessarily Defined Function	QA	Resolved

CONTENTS

8. Findings

8.1. High Findings

[H-01] Commission Contract Validation Leads to Locked Rewards

Description

The `rewards-v3` contract stores the PoX rewards for a specific cycle. Once the cycle concludes, the `process-rewards` function transfers the reward STX and sBTC to the `reserve-v3` contract. This function requires five parameters, four of which are permissioned protocol contracts:

- `commission-ststx-contract` - The current whitelisted stSTX commission contract implementation.
- `commission-ststxbtc-contract` - The current whitelisted stSTXBTC commission contract implementation.
- `staking-contract` - The current whitelisted staking contract implementation.
- `reserve` - The current whitelisted reserve contract implementation.

All four parameters are validated as whitelisted protocol contracts through the `dao.check-is-protocol` function.

```
(try! (contract-call? .dao check-is-enabled))
(try! (contract-call? .dao check-is-protocol (contract-of reserve)))
(try! (contract-call? .dao check-is-protocol
  (contract-of commission-ststx-contract)))
(try! (contract-call? .dao check-is-protocol
  (contract-of commission-ststxbtc-contract)))
(try! (contract-call? .dao check-is-protocol (contract-of staking-contract)))
```

The issue arises because both commission contracts are only validated as inheriting the `<commission-trait>` trait and as being protocol contracts. This creates a problem, as the values passed for `commission-ststx-contract` and `commission-ststxbtc-contract` can be swapped or duplicated.

Mixing up the stSTX and stSTXBTC commission addresses or passing the same address to both will permanently lock the entire cycle reward amount for either token, thereby disrupting core protocol functionality and resulting in the loss of all stacker funds for the given reward cycle.

CONTENTS

Recommendation

The simplest solution to this issue is to split the `<commission-trait>` into two new traits: `<ststx-commission-trait>` and `<ststxbtc-commission-trait>`, and use each respectively as the accepted trait type for the two parameters.

8.2. Medium Findings

[M-01] stSTX Price Vulnerable to Manipulation with Flashloan Access to Entire Supply

Description

The stSTXBTC upgrade includes a feeless swap contract, `swap-ststx-ststxbtc-v1`, which facilitates atomic swaps between stSTX and stSTXBTC. The contract offers two public functions:

- stSTX to stSTXBTC (`swap-ststx-for-ststxbtc`)
- stSTXBTC to stSTX (`swap-ststxbtc-for-ststx`)

Both functions operate by burning the input token amount and minting a proportionate amount of the output token.

The issue arises from the pricing mechanism of stSTX within the `get-stx-per-ststx` function. This function employs a helper with the following condition:

```
(if (is-eq ststx-supply u0)
        DENOMINATOR_6
        (/ (* stx-amount DENOMINATOR_6) ststx-supply)
)
```

Thus, theoretically, if one gains access to the entire supply of stSTX through a flashloan, it would allow the STX:stSTX rate to be reset to 1:1, potentially causing significant economic disruption.

Recommendation

Consider implementing a minimum supply threshold for each token to remain after such a swap.

CONTENTS

[M-02] Semi-Trusted Position-Managing Contracts Lack Balance Validation

Description

Positions (`position-trait`) are external stSTXBTC token managers intended to oversee users' direct staking while still allowing users to claim their sBTC rewards directly through the holder <-> position accounting in `ststxbtc-tracking` .

These contracts are designed as semi-trusted actors because they are permitted to provide token balance information directly to the core contracts, which is then used as the state for reward calculation.

The issue with the current setup of position contracts is that their `get-balance` output is never validated. They only need to be whitelisted within the `supported-positions` mapping to function in this capacity.

Whenever `refresh-position` is called on a holder with a whitelisted position contract, the output of the position's value is not validated against the holdings of the position contract, nor is it checked against the total supply of stSTXBTC.

```
(position-balance (try! (contract-call? position get-balance holder)))
```

Although the above is whitelisted, any vulnerability or manipulation vector discovered within it could jeopardize sBTC rewards for the entire pool of stSTXBTC holders.

Recommendation

Consider adding an additional accounting layer in `ststxbtc-tracking-data` , which is responsible for tracking the portion of the position contracts' holder balances to ensure they do not exceed the contract's actual stSTXBTC holdings when `refresh-position` is called.

[M-03] Positions That Are Subsequently Un-Whitelisted Can Replay Reward Claims

Description

The function `ststxbtc-tracking::get-pending-rewards` returns 0 if the `holder` is also whitelisted as a position. This measure is taken because whenever stSTXBTC is minted or transferred to a position contract, the `refresh-wallet` mechanism is directly invoked on the said position contract. This results in double accounting, necessitating this measure.

Occurrences in `ststxbtc-token::mint-for-protocol` , `ststxbtc-token::burn-for-protocol` , and `ststxbtc-token::transfer` :

```
(try! (contract-call? .ststxbtc-tracking refresh-wallet sender
  (ft-get-balance ststxbtc sender)))
```

The issue arises because positions in position contracts can still be claimed if those contracts are removed from the whitelist. The only safeguard against such claims is the following logic in `ststxbtc-tracking::get-pending-rewards` :

```
(is-holder-position
  (contract-call? .ststxbtc-tracking-data get-supported-positions holder))

(if is-holder-position
    (ok u0)
    (ok rewards)
)
```

However, since a position that is no longer whitelisted does not have its `cumm-reward` set to the current value, anyone can call balance erased or its `ststxbtc-tracking::claim-pending-rewards` with the position contract as `holder` , effectively locking sBTC away from actual stakers.

Recommendation

Consider implementing conditional logic in `refresh-wallet` to prevent the `holder-position` mapping entry for the position contract from being set while it is a whitelisted position. Additionally, consider invoking `update-holder-position-amount` on a contract in `set-supported-positions` when removing it from the whitelist.

CONTENTS

[M-04] Holders in Non-Whitelisted Positions Could Continue Claiming sBTC Rewards

Description

stSTXBTC holders can collect the sBTC rewards generated by their holdings even when these holdings are deployed in another protocol, thanks to the position accounting mechanism in `ststxbtc-tracking`.

Protocols that are integrated and whitelisted into the position mechanism must provide an interface for the stSTXBTC balances of users they manage. This allows users to claim the rewards they are entitled to, facilitated through `position-trait`.

The issue arises because the balances of users in external protocols are only verified in `refresh-position`, where the position contract is checked for being whitelisted. However, other parts of the process simply refer back to the `holder-position` mapping without verifying if the position remains whitelisted. As a result, users can continue to claim sBTC rewards every two weeks based solely on their `holder-position` entry until they are manually removed from the mapping through `set-holder-position` by a privileged account.

Recommendation

Consider implementing a check to ensure the strategy is still in the `supported-positions` whitelist when calling `claim-pending-rewards`

This approach might lead to situations where users are unfairly separated from rewards they have not yet claimed. To address this edge case, introduce a mapping to track the `cumm-reward` value at which a position is excluded from the whitelist.

8.3. Low Findings

[L-01] Position Input Not Validated as `position-trait`

Description

For positions to be utilized within the position accounting system, they must adhere to the `position-trait` as the `refresh-position` function requires a `<position-trait>::get-balance` for the position's holder.

The following entry points accept a position as a `principal` :

- `ststxbtc-tracking-data`
  - `set-supported-positions`
  - `set-holder-position`
  - `update-holder-position`
  - `update-holder-position-amount`
- `ststxbtc-tracking`
  - `get-pending-rewards`
  - `claim-pending-rewards`

Recommendation

Consider changing the `principal` input type to `<position-trait>` in all the above instances to prevent a corrupted state.

CONTENTS

8.4. QA Findings

[QA-01] Incorrect Parameter Naming

Description

The parameter for the stSTXBTC amount intended for withdrawal in the `init-withdraw` function is incorrectly named `stx-amount`.

```
(define-public (init-withdraw
  (reserve <reserve-trait>)
  (direct-helpers <direct-helpers-trait>)
  (stx-amount uint)
)
...
)
```

Recommendation

It is recommended to rename the parameter to `ststxbtc-amount`.

CONTENTS

[QA-02] Unnecessarily Defined Function

Description

The function `stacking-dao-core-btc-v1::get-withdraw-unlock-burn-height` is redundant as it is identical to `stacking-dao-core-v4::get-withdraw-unlock-burn-height`.

```
(define-read-only (get-withdraw-unlock-burn-height)
  (let (
    (current-cycle (current-pox-reward-cycle))
    (start-block-next-cycle (reward-cycle-to-burn-height (+ current-cycle u1)))
    (withdraw-offset
      (contract-call? .data-core-v1 get-cycle-withdraw-offset))
    (withdraw-inset (contract-call? .data-core-v2 get-cycle-withdraw-inset))
  )
    (if (< burn-block-height (- start-block-next-cycle withdraw-offset))
      ;; Can withdraw next cycle
      (ok (+ withdraw-inset start-block-next-cycle))

      ;; Withdraw cycle after next
      (ok (+ withdraw-inset start-block-next-cycle (get-reward-cycle-length)))
    )
  )
)
```

Recommendation

It is advisable to re-implement the function as a wrapped call to `stacking-dao-core-v4::get-withdraw-unlock-burn-height`.