ClarityAlliance

SIP-31 BOOT CONTRACT SECURITY REVIEW

Conducted by:
KRISTIAN APOSTOLOV, ALIN BARBATEI (ABA)

JULY 25TH, 2025

CONTENTS

ClarityAlliance
Security Review

SIP-31 Boot Contract

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.

CONTENTS

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 security review of SIP-31 Boot Contract, a component of the Stacks blockchain protocol, where Clarity Alliance reviewed the scope and provided insights on improving the protocol.

4. About Stacks L2

Stacks is a layer-2 blockchain that uses Bitcoin as a base layer for security and enables decentralized apps and predictable smart contracts using the Clarity language. Stacks implements Proof of Transfer (PoX) mining that anchors to Bitcoin security. Leader election happens at the Bitcoin blockchain and Stacks (STX) miners write new blocks on the separate Stacks blockchain. With PoX there is no need to modify Bitcoin to enable smart contracts and decentralized apps.

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

Scope

The following contracts were in the scope of the security review:

- `stackslib/src/chainstate/stacks/boot/sip-031.clar`

Initial Commit Reviewed:
41f7146cb28eb1ce1e4b1ce4b51c5cd84e5ed1fb

Final Commit After Remediations:
205986c666e6dfd39760b9f6073964d6ede2364b

ClarityAlliance
Security Review

SIP-31 Boot Contract

CONTENTS

7. Executive Summary

Over the course of the security review, Kristian Apostolov, Alin Barbatei (ABA) engaged with - to review Stacks L2. In this period of time a total of **8** issues were uncovered.

Protocol Summary

Protocol Name	Stacks L2
Date	July 25th, 2025

Findings Count

Severity	Amount
Medium	1
Low	1
QA	6
Total Findings	**8**

CONTENTS

Summary of Findings

ID	Title	Severity	Status
[M-01]	Incorrect Calculation of Claimed Vested Amount	Medium	Resolved
[L-01]	Potential SIP-31 Contradiction in the get-recipient Function	Low	Resolved
[QA-01]	Ambiguous Reversion in calc-claim-able-amount with Invalid burn-height	QA	Resolved
[QA-02]	Missing Event Emission on Important Actions	QA	Resolved
[QA-03]	Constants Should Be Uppercase	QA	Resolved
[QA-04]	Contract Comments Uniformity	QA	Resolved
[QA-05]	Remove Redundant Begin Block	QA	Resolved
[QA-06]	Add Network Validation for New Recipient	QA	Resolved

8. Findings

8.1. Medium Findings

[M-01] Incorrect Calculation of Claimed Vested Amount

Description

When claiming vested STX through the `sip-031::claim` contract, the total vested and claimed amount is updated in the `vested-claimed-amount` variable as follows:

```
;; let declarations

(total-vested (calc-total-vested burn-block-height))
(vested-claimed (var-get vested-claimed-amount))

;; ... code ...

(var-set vested-claimed-amount (+ vested-claimed total-vested))
```

This approach is incorrect because the `total-vested` amount, obtained by calling `calc-total-vested`, represents the total released or matured tokens from the contract's deployment up to the current point. The `total-vested` amount is already the correct value to store. By adding the previously saved vested and claimed amount, double counting occurs, causing the `vested-claimed-amount` to become significantly inflated.

This issue affects all third-party entities that rely on the current claimed/vested amounts via the `sip-031::get-vested-claimed-amount` function. It also leads to call reverts when executing `sip-031::calc-claimed-amount` due to an underflow error when subtracting the inflated already-claimed-vested amount from the newly calculated vested amount:

```
(- total-vested (var-get vested-claimed-amount))
```

Recommendation

Set the `vested-claimed-amount` variable in the `sip-031::claim` function to the `total-vested` variable.

Alternatively, consider removing the `vested-claimed-amount` variable and modifying the `calc-total-vested` function to closely align with the `claim` logic.

Note: This issue was also identified by the development team during the audit.

CONTENTS

8.2. Low Findings

[L-01] Potential SIP-31 Contradiction in the get-recipient Function

Description

According to the current SIP-31 document draft, the SIP-31 boot contract will include a function to retrieve the current recipient of the emissions as follows:

> A read-only function get-recipient returning the current principal (set with update-recipient).

This section indicates that the `get-recipient` function solely returns the current principal.

However, in the subsequent code block that illustrates the functions, `get-recipient` is described as returning a Response type value (`Returns (response principal uint)`) with a `uint` error code in case of failure.

```
(define-read-only (get-recipient) ...)) ;; Returns (response principal uint)
```

The current implementation aligns with the initial description of `get-recipient` (which is also simpler for integrating parties) and only returns the recipient principal.

```
(define-read-only (get-recipient) (var-get recipient))
```

This inconsistency may lead to confusion for external integrators who expect the `sip-031::get-recipient` function to return as specified in the second part of the SIP technical implementation.

Recommendation

Having a response type for the `get-recipient` function is unnecessary. We recommend maintaining the current implementation but modifying the `sips/sip-031/sip-031.md#technical-implementation` document to reflect that the `get-recipient` function only returns a principal.

8.3. QA Findings

[QA-01] Ambiguous Reversion in `calc-claimable-amount` with Invalid `burn-height`

Description

The `sip-031::calc-claimable-amount` function allows users to input any arbitrary `burn-height` parameter, which serves as a proxy for the burn-block-height at which the claimable amount is simulated. Currently, there is no validation for the `burn-height` value. If users inadvertently provide a value below the burn height of 907740 (`deploy-block-height`), it results in an underflow in the `calc-total-vested` function.

This lack of validation complicates debugging for any third-party or external integrator.

Recommendation

In the `sip-031::calc-claimable-amount` function, ensure that the `burn-height` is at least equal to the `deploy-block-height`. If it is not, either revert with a custom, meaningful error or return `0` as a logically valid response.

Example implementation:

```
(define-read-only (calc-claimable-amount (burn-height uint))
    (if (< burn-height deploy-block-height)
        u0
        (let
            (
                (total-vested (calc-total-vested burn-height))
                (reserved (- INITIAL_MINT_AMOUNT total-vested))
                (balance (stx-get-balance (as-contract tx-sender)))
                (claimable
                    (if  (> balance reserved)
                         (- balance reserved)
                         u0))
            )
            claimable
        )
    )
)
```

CONTENTS

[QA-02] Missing Event Emission on Important Actions

Description

The `sip-031` contract lacks event emissions, specifically `print` statements, for any actions. This absence complicates the ability of off-chain monitoring systems to track contract activities effectively.

Recommendation

Incorporate `print` statements with pertinent details in the `update-recipient` function (including the old recipient and new recipient) and in the `claim` function (including the contract caller and claimable amount).

CONTENTS

[QA-03] Constants Should Be Uppercase

Description

Typically, constants are written in full uppercase. In the `sip-031` contract, all constants adhere to this convention except for one: `deploy-block-height` .

Recommendation

To enhance code readability, rename the `deploy-block-height` constant to `DEPLOY_BLOCK_HEIGHT` and position it alongside the other constants, rather than placing it after a data variable.

[QA-04] Contract Comments Uniformity

Description

Within the `sip-031` contract, there are opportunities to enhance the existing comments:

1. The comment for the `calc-claimable-amount` function states:

   > Returns the amount of STX that is claimable from the vested balance at burn-height

However, this information is incomplete. It should also consider the current STX contract balance, which may result in a different claimable amount at the specified burn-height when `claim` is called now, compared to the response from a `calc-claimable-amount` call made in the past.

2. Extra whitespaces and newlines can be removed:
   - In the `calc-total-vested` function, one of the internal comments has two whitespaces between `unlocked` and `This avoids` instead of one.
   - The description comments for the `update-recipient` function contain empty `;;` lines between the three fully commented lines.

3. Unlike other boot contracts (e.g., `costs-*`, `pox-*`, `signers-voting`), which include an initial header-type comment, the `sip-031` contract lacks such a comment.

To enhance contract uniformity, the aforementioned comment improvements should be implemented.

Recommendation

To better describe the functionality of the `calc-claimable-amount` function, add an additional comment that explains the STX balance behavior. For example:

```
;;
   Returns the amount of STX that is claimable from the vested balance at `burn-height`
;;
   while considering the current STX contract balance; May differ from the actual `claim` call
```

For the extra whitespaces and newlines, remove the excess.

Additionally, consider adding a brief header-type comment to describe the contract's purpose.

CONTENTS

[QA-05] Remove Redundant Begin Block

Description

In the `sip-031::update-recipient` function, a `begin` block is unnecessarily nested within another `begin` block.

```
(define-public (update-recipient (new-recipient principal))
  (begin
    (begin
    ;; ... code ...
```

This redundancy slightly increases execution costs and reduces code readability.

Recommendation

Eliminate one of the `begin` blocks from the `sip-031::update-recipient` function.

ClarityAlliance
Security Review

SIP-31 Boot Contract

CONTENTS

[QA-06] Add Network Validation for New Recipient

Description

In the `sip-031` contract, when updating the recipient of STX tokens, there is currently no validation to ensure that the principal is standard, specifically belonging to the current network. This oversight could lead to the accidental use of a testnet principal instead of a mainnet principal, which would render the contracts unusable.

Recommendation

Ensure that the `new-recipient` principal is valid for the current network by using the `is-standard` function.

Note: This improvement was mentioned by the development team during the audit.