ClarityAlliance

GRANITE (UPGRADE V3) SECURITY REVIEW

Conducted by:
KRISTIAN APOSTOLOV, ALIN BARBATEI (ABA)

DECEMBER 2ND, 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 claritalliance.org.

ClarityAlliance
Security Review

Granite
(Upgrade v3)

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 time-boxed security review of Granite Protocol, where Clarity Alliance reviewed the scope and provided insights on improving the protocol.

4. About Granite

Granite is a Bitcoin Liquidity Protocol that provides the first truly non-custodial, secure, and decentralized way to borrow against Bitcoin.

The protocol allows borrowers to take stablecoin loans using Bitcoin as collateral, without exposure to counterparty or rehypothecation risk. Liquidity providers can earn yield on stablecoins by providing liquidity to the pool, which is then lent to borrowers.

Loans in Granite are best thought of as lines of credit, without set terms or repayment schedules. As long as the borrower maintains an adequate loan-to-value ratio (LTV), keeping their account in good health, they are not subject to liquidation. If a borrower's LTV falls too low, a portion of their capital will be liquidated to bring their account back to solvency.

Granite enables BTC users to access DeFi without centralized custodians by leveraging Stacks' Nakamoto upgrade and sBTC Bitcoin bridge.

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, provided across two pull requests in the core-v1 repository, were in scope of the security review. The scope of this review consisted of the code updates introduced in the referenced pull requests:

- `flash-loan-v1.clar`
- `governance-v1.clar`
- `meta-governance-v1.clar`

Initial Commit Reviewed:
- PR #35 — Commit 80c43136675b25c366065cde2261c18e91cfc003

Intermediate Commit Reviewed:
- PR #40 — Commit 32d755d25784e064087640f7bd5b17d1ccd2d572

Final Commit After Remediations:
c5fa1509c5696ca57035fdd58f8c1a4a639ff740

CONTENTS

7. Executive Summary

Over the course of the security review, Kristian Apostolov, Alin Barbatei (ABA) engaged with - to review Granite. In this period of time a total of **4** issues were uncovered.

Protocol Summary

Protocol Name	Granite
Date	December 2nd, 2025

Findings Count

Severity	Amount
Low	2
QA	2
Total Findings	**4**

CONTENTS

Summary of Findings

ID	Title	Severity	Status
[L-01]	Flash Loan Fee Cannot Be Waived	Low	Resolved
[L-02]	Meta Governance Initialization Skips Crucial Duplication Checks	Low	Resolved
[QA-01]	Governance Guardian Operation Status Not Verified	QA	Resolved
[QA-02]	Miscellaneous Governance Contract Improvements	QA	Resolved

8. Findings

8.1. Low Findings

[L-01] Flash Loan Fee Cannot Be Waived

Description

The current logic in the `flash-loan-v1::flash-loan` function always assumes the presence of a flash loan fee and attempts to transfer it to governance:

```
(try! (contract-call? .mock-usdc transfer flash-loan-fee caller
  (contract-call? .state-v1 get-governance) none))
```

Governance has the ability to set the flash loan fee to any arbitrary value. However, because the fee is always assumed to exist, governance cannot actually set it to 0. This could be beneficial under certain market conditions. Instead, they are forced to use a minimal value, which unnecessarily incurs additional execution costs.

Recommendation

In the `flash-loan-v1::flash-loan` function, ensure that the `flash-loan-fee` is greater than 0 before attempting to initiate the transfer.

[L-02] Meta Governance Initialization Skips Crucial Duplication Checks

Description

In the `meta-governance-v1` contract, the `initialize-governance` function can be called post-deployment to set up the initial group of governance multisig holders, allowing for a list of up to 5 members. Each principal provided is processed by the `set-governance-multisig` function, which increments the count and calls `add-governance-multisig` to finalize the new multisig.

However, the `add-governance-multisig` function does not verify whether a governance member already exists, potentially leading to duplicate entries. This oversight can render the governance unusable, as the voting threshold may become unattainable.

Recommendation

Consolidate all validations and logic from each branch in `execute-update-governance-multisig` into dedicated functions.

For instance, when adding a multisig member, incorporate the `(asserts! (not (is-already-member governance)) ERR-ALREADY-GOVERNANCE-MEMBER)` check and the `(var-set governance-accounts-count (+ (var-get governance-accounts-count) u1))` increment directly within the `add-governance-multisig` function.

Example of `add-governance-multisig`:

```
(define-private (add-governance-multisig (governance principal))
  (begin
    (asserts! (not
      (is-already-member governance)) ERR-ALREADY-GOVERNANCE-MEMBER)
    (map-set governance-accounts governance true)
    (var-set governance-accounts-count (+
      (var-get governance-accounts-count) u1))
    (print {
      action: “add-governance-multisig”,
      governance: governance
    })
    SUCCESS
))
```

In this scenario, the `set-governance-multisig` function should transition from using `map` logic to `fold`, ensuring that any error encountered is propagated, causing the entire sequence to revert. Additionally, the revised `set-governance-multisig` should not increment the `governance accounts-count` variable again.

For `remove-governance-multisig`, consolidating all checks within it would enhance uniformity, even if not strictly necessary for functionality.

8.2. QA Findings

[QA-01] Governance Guardian Operation Status Not Verified

Description

The `governance-v1` contract employs two functions, `add-guardian` and `remove-guardian`, to manage guardians. These functions are executed directly with the provided proposal input.

```
(define-private (add-guardian (guardian principal))
  (begin
    (map-set guardians guardian true)
    (print {
      action: “add-guardian”,
      guardian: guardian
    })
    SUCCESS
))

(define-private (remove-guardian (guardian principal))
  (begin
    (map-delete guardians guardian)
    (print {
      action: “remove-guardian”,
      guardian: guardian
    })
    SUCCESS
))
```

However, there is no verification during execution to ensure that a new guardian is being added or an existing guardian is being removed. Calls made with invalid guardians will proceed without any clear indication of failure.

Recommendation

Replace the use of `map-set` with `map-insert` and print the result of `map-insert` and `map-delete` in each function as a status indicator. The `map-insert` will return true if the guardian was not previously in the mapping, and `map-delete` will return true if the guardian was present before deletion.

Further validation can be achieved by asserting the return status of `map-insert` and `map-delete`, rather than merely printing. Additional checks can be incorporated in `initiate-proposal-to-update-guardians`. However, initializing a new governance via `initialize-governance` would bypass these checks when using `set-guardians`.

Note that if assertions are used, printing is unnecessary, as successful execution would only occur if no invalid guardian was set.

[QA-02] Miscellaneous Governance Contract Improvements

Description

The `governance-v1` contract has a few minor improvements that can be made:

- There is a typo at line 614 where the word `excuted` should be corrected to `executed`.
- The `initialize-governance` function currently prints an event with the `action` field set to `meta-governance` and another field named `meta-governance` with the value of the currently paired governance contract `.meta-governance-v1`:

```
(print {
    action: "meta-governance",
    meta-governance: .meta-governance-v1
})
```

In this context, consider whether changing the `action` value to `initialize-governance` would be more appropriate. Additionally, include the `guardians-addrs` in the print statement.

Recommendation

Implement the suggested improvements.