ClarityAlliance

GRANITE SECURITY REVIEW

Conducted by:
KRISTIAN APOSTOLOV, Stormy

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
[H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
[M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
[M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
[M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
[M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
[L-01] Rounding Interest Against Protocol 15
[L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
[QA-01] Liquidation Flag Setter Can Be Improved 17
[QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
[QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
[QA-04] Market Asset Value Should Be Rounded Up 20
[QA-05] Simplify socialize-bad-debt 21

ClarityAlliance
Security Review
Granite

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

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
[H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
[M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
[M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
[M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
[M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
[L-01] Rounding Interest Against Protocol 15
[L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
[QA-01] Liquidation Flag Setter Can Be Improved 17
[QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
[QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
[QA-04] Market Asset Value Should Be Rounded Up 20
[QA-05] Simplify socialize-bad-debt 21

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

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
[H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
[M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
[M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
[M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
[M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
[L-01] Rounding Interest Against Protocol 15
[L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
[QA-01] Liquidation Flag Setter Can Be Improved 17
[QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
[QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
[QA-04] Market Asset Value Should Be Rounded Up 20
[QA-05] Simplify socialize-bad-debt 21

3. Introduction

A time-boxed security review of the Granite implementation, where Clarity Alliance reviewed the scope, whilst simultaneously building out a testing suite for the protocol.

4. About Granite

The Granite Protocol is an autonomous Bitcoin liquidity protocol where users can participate as liquidity providers, borrowers, or liquidators.

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' soon-to-be-launched Nakamoto upgrade and sBTC Bitcoin bridge.

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

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
 5.1. Impact 4
 5.2. Likelihood 5
 5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
 [H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
 [M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
 [M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
 [M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
 [M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
 [L-01] Rounding Interest Against Protocol 15
 [L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
 [QA-01] Liquidation Flag Setter Can Be Improved 17
 [QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
 [QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
 [QA-04] Market Asset Value Should Be Rounded Up 20
 [QA-05] Simplify socialize-bad-debt 21

ClarityAlliance
Security Review
Granite

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

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
[H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
[M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
[M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
[M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
[M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
[L-01] Rounding Interest Against Protocol 15
[L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
[QA-01] Liquidation Flag Setter Can Be Improved 17
[QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
[QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
[QA-04] Market Asset Value Should Be Rounded Up 20
[QA-05] Simplify socialize-bad-debt 21

ClarityAlliance
Security Review
Granite

6. Security Assessment Summary

Review Commit Hash:
9c3ffb40858a005e292de432582188c3d3f69289

- `contracts/liquidator.clar`
- `contracts/borrower.clar`
- `contracts/liquidity-provider.clar`
- `contracts/governance.clar`
- `contracts/state.clar`
- `contracts/meta-governance.clar`
- `contracts/traits/trait-sip-010.clar`
- `contracts/modules/pyth-oracle.clar`
- `contracts/modules/math.clar`
- `contracts/modules/linear-kinked-ir.clar`
- `contracts/staking.clar`
- `contracts/staking-reward.clar`

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
[H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
[M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
[M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
[M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
[M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
[L-01] Rounding Interest Against Protocol 15
[L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
[QA-01] Liquidation Flag Setter Can Be Improved 17
[QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
[QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
[QA-04] Market Asset Value Should Be Rounded Up 20
[QA-05] Simplify socialize-bad-debt 21

7. Executive Summary

Over the course of the security review, Kristian Apostolov, Stormy engaged with Trust Machines to review Granite. In this period of time a total of **12** issues were uncovered.

Protocol Summary

Protocol Name	Granite
Protocol Type	Lending Market

Findings Count

Severity	Amount
High	1
Medium	4
Low	2
QA	5
Total Findings	**12**

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
[H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
[M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
[M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
[M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
[M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
[L-01] Rounding Interest Against Protocol 15
[L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
[QA-01] Liquidation Flag Setter Can Be Improved 17
[QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
[QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
[QA-04] Market Asset Value Should Be Rounded Up 20
[QA-05] Simplify socialize-bad-debt 21

ClarityAlliance
Security Review
Granite

Summary of Findings

ID	Title	Severity	Status
[H-01]	Not Calling accrue-interest Before a Safety Module Balance Mutation	High	Resolved
[M-01]	Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position	Medium	Resolved
[M-02]	Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero	Medium	Resolved
[M-03]	last-accrued-block Not Set When Mutating interest-accrual-enabled	Medium	Resolved
[M-04]	Bad Debt Socialization Race Condition	Medium	Resolved
[L-01]	Rounding Interest Against Protocol	Low	Resolved
[L-02]	Incorrect Interest Read-Only Output	Low	Resolved
[QA-01]	Liquidation Flag Setter Can Be Improved	QA	Resolved
[QA-02]	Duplicated Functions for Fetching Collateral Parameters Can Be Merged	QA	Resolved
[QA-03]	Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed	QA	Resolved
[QA-04]	Market Asset Value Should Be Rounded Up	QA	Resolved
[QA-05]	Simplify socialize-bad-debt	QA	Resolved

CONTENTS
1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
[H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
[M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
[M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
[M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
[M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
[L-01] Rounding Interest Against Protocol 15
[L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
[QA-01] Liquidation Flag Setter Can Be Improved 17
[QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
[QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
[QA-04] Market Asset Value Should Be Rounded Up 20
[QA-05] Simplify socialize-bad-debt 21

8. Findings

8.1. High Findings

[H-01] Not Calling `accrue-interest` Before a Safety Module Balance Mutation

Description

The `accrue-interest` function, which is replicated across most entry point contracts in the system, calculates the accumulated interest from the last accrual to the present moment. This interest is then divided among three distinct groups for OI bookkeeping:

- **LP**: The portion containing interest owed to ordinary LPs.
- **Staked**: Owed to LPs who opted into the safety module, thus acting as insurers of the system in cases of bad debt accrual.
- **Protocol**: OI attributed to the protocol reserve.

The staking part of the interest is calculated with the following formula:

```
(/ (* lp-interest staking-reward-percentage) one-8)
```

where `lp-interest` is `total-interest - protocol-interest`, and `staking-reward-percentage` is a percentage calculated through a linear slope based on a "utilization" ratio, similar to how ordinary interest is calculated. In this case, the ratio used is `staked-lp-tokens / lp-tokens`.

Contrary to the general interest slope, the one used in staking reward calculations is inverted to incentivize a healthy `staked-lp-tokens / lp-tokens` balance in the system.

The issue arises because the protocol does not compound interest before mutating staked token balances through operations like `stake` and `initiate-stake`.

This poses a problem for the system's interest calculations as it allows for parameters controlling the OI amounts distributed to ordinary LPs and staked ones to be manipulated.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
   5.1. Impact 4
   5.2. Likelihood 5
   5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
[H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
[M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
[M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
[M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
[M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
[L-01] Rounding Interest Against Protocol 15
[L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
[QA-01] Liquidation Flag Setter Can Be Improved 17
[QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
[QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
[QA-04] Market Asset Value Should Be Rounded Up 20
[QA-05] Simplify socialize-bad-debt 21

The above issue could be exploited in one of two scenarios:

- **Minting Staked Tokens**: Sandwiching an accrual event by minting an amount of staked tokens to extract interest that belonged to other stakers in the system who were present for those blocks, thereby lowering the percentage of accrued interest they are supposed to receive.
- **Burning Staked Tokens:** Sandwiching an accrual event by burning an amount of staked tokens to avoid reaching as deeply into the second slope of reward calculation, thus receiving a slightly lower amount of tokens in return for providing significantly less security for the protocol's bad debt mechanism.

Recommendation

Consider implementing a copy of `accrue-interest` in `staking.clar` and calling it before `stake`, `unstake`, and `reconcile-lp-token-balance`.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
[H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
[M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
[M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
[M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
[M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
[L-01] Rounding Interest Against Protocol 15
[L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
[QA-01] Liquidation Flag Setter Can Be Improved 17
[QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
[QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
[QA-04] Market Asset Value Should Be Rounded Up 20
[QA-05] Simplify socialize-bad-debt 21

ClarityAlliance
Security Review
Granite

8.2. Medium Findings

[M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position

Description

The private function `remove-user-collateral` is called by `remove-collateral` when a user wants to retrieve some of the collateral they initially added to the protocol. The purpose of `remove-user-collateral` is to correctly account for the storage variables when either a partial or full collateral is being withdrawn.

For security reasons, a governance feature called `remove-collateral-enabled` can temporarily restrict the removal of collateral from the contracts. However, unlike `update-user-collateral`, this security check is not implemented in the function `update-remove-collateral`. As a result, even if the removal of collateral is temporarily disabled by the governance, collateral can still be removed from the contract by fully withdrawing a collateral position.

Recommendation

The removal of any collateral from the contracts should not be allowed when `remove-collateral-enabled` is temporarily disabled by the governance.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
[H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
[M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
[M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
[M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
[M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
[L-01] Rounding Interest Against Protocol 15
[L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
[QA-01] Liquidation Flag Setter Can Be Improved 17
[QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
[QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
[QA-04] Market Asset Value Should Be Rounded Up 20
[QA-05] Simplify socialize-bad-debt 21

[M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero

Description

Users are allowed to add up to 10 different collaterals simultaneously to top up their user positions' `max-ltv`. When a position's collateral amount drops to zero after removing collateral, the system updates the user's collateral list by removing the unused collateral from it.

By examining the `liquidate-collateral` function, we can see that the system adjusts the amount of collateral in `user-collaterals` accordingly. However, the system does not update the user's collateral list when `user-collaterals` drops to zero.

Failing to correctly update the user's collateral list and remove the unused collateral may restrict the user's ability to add another token as collateral due to the enforced collateral limit per user.

Recommendation

We recommend updating the user's collateral list once the liquidation reduces a position amount to zero. Otherwise, there won't be a way to remove it afterward, which may affect the user's ability to add another collateral for use due to overflow when hitting the collateral list limit.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
[H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
[M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
[M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
[M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
[M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
[L-01] Rounding Interest Against Protocol 15
[L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
[QA-01] Liquidation Flag Setter Can Be Improved 17
[QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
[QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
[QA-04] Market Asset Value Should Be Rounded Up 20
[QA-05] Simplify socialize-bad-debt 21

[M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled

Description

The function `state::set-interest-accrual-enabled` and the market pausing flow it is used in are responsible for pausing a market in extraordinary scenarios. The issue arises because `last-accrued-block` is not set alongside `interest-accrual-enabled` in `state::set-interest-accrual-enabled`. Disabling interest accrual is likely to be followed by interest parameter mutations before re-enabling the market's interest accrual. If `last-accrual-block` is not set to the current block, interest will be accrued for the period during which it should have been disabled.

This situation could lead to unexpected liquidations, even if the liquidation warmup has been adjusted accordingly, because users' health factors will drastically decrease when any of the `accrue-interest` functions across the system are invoked.

Recommendation

Consider setting `last-accrued-block` to `block-height` in `state::set-interest-accrual-enabled`.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
[H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
[M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
[M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
[M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
[M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
[L-01] Rounding Interest Against Protocol 15
[L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
[QA-01] Liquidation Flag Setter Can Be Improved 17
[QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
[QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
[QA-04] Market Asset Value Should Be Rounded Up 20
[QA-05] Simplify socialize-bad-debt 21

[M-04] Bad Debt Socialization Race Condition

Description

The safety module in the system plays a critical role in addressing situations where bad debt occurs due to undercollateralized loans.

The issue arises from how the system queues unstaking requests within the safety module. Specifically, it calculates and saves the number of LP tokens a user is entitled to receive after the waiting period ends via the `initiate-unstake` function.

```
(lp-tokens-to-return (convert-to-lp-tokens staked-lp-tokens false))
```

This process can inadvertently encourage stakers to frontrun bad debt liquidation events by calling `initiate-unstake` beforehand, ensuring they avoid the impending debt socialization.

Recommendation

Addressing this behavior would require significant code changes and a redesign of the staking module. Therefore, it may be practical to acknowledge this limitation within the system and closely monitor staker activity during bad debt liquidations to mitigate potential issues.

Granite Team: implemented a withdrawal shares accounting system to socialize bad debt across both active stakers and those who are unstaking.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
[H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
[M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
[M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
[M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
[M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
[L-01] Rounding Interest Against Protocol 15
[L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
[QA-01] Liquidation Flag Setter Can Be Improved 17
[QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
[QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
[QA-04] Market Asset Value Should Be Rounded Up 20
[QA-05] Simplify socialize-bad-debt 21

8.3. Low Findings

[L-01] Rounding Interest Against Protocol

Description

The protocol calculates how a loan repayment should be split based on the ratio of principal to interest of the given position using the following:

```
(interest-part (/ (* interest-accrued repay-amount) current-debt))
(principal-part (- repay-amount interest-part))
```

The issue here is that `interest-part` is determined through round-down division, which is unfavorable for the protocol. Rounding down the interest increases the amount of principal being repaid, thereby reducing the amount of interest generated from keeping the position active.

Recommendation

Consider using round-up division in `math.clar::calculate-interest-portions` .

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
[H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
[M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
[M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
[M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
[M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
[L-01] Rounding Interest Against Protocol 15
[L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
[QA-01] Liquidation Flag Setter Can Be Improved 17
[QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
[QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
[QA-04] Market Asset Value Should Be Rounded Up 20
[QA-05] Simplify socialize-bad-debt 21

[L-02] Incorrect Interest Read-Only Output

Description

The linear interest component of the protocol is calculated using a variable slope mechanism, which incorporates a kink target and a constant base rate.

The issue arises because the `base-ir` is also added when `OI == 0`.

Although `base-ir` is generally applicable and should be included in most cases, it should not be added when there are no borrows in the system.

Since accrued interest is the product of the accumulated interest and the current OI, this does not impact the protocol's accounting. However, it does cause `get-ir` to output an invalid value, potentially leading to confusion when reading it.

Recommendation

Consider returning early in the case of `util-with-res == 0` in `linear-kinked-ir.clar::get-ir`:

```
(define-private (ir-calc (util-with-res uint))
  (if (is-eq util-with-res u0)
    u0
    (if
      (>= util-with-res (var-get utilization-kink))
       (interest-util-geq-kink util-with-res)
       (interest-util-less-than-kink util-with-res)
    )
  )
)
```

CONTENTS
1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
[H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
[M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
[M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
[M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
[M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
[L-01] Rounding Interest Against Protocol 15
[L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
[QA-01] Liquidation Flag Setter Can Be Improved 17
[QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
[QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
[QA-04] Market Asset Value Should Be Rounded Up 20
[QA-05] Simplify socialize-bad-debt 21

8.3. QA Findings

[QA-01] Liquidation Flag Setter Can Be Improved

Description

In the protocol, the `set-liquidation-flag` is a governance function used to enable or disable the liquidation logic and set an appropriate liquidation `cooldown` when needed. Currently, we use an input variable cooldown which represents a specific block until which liquidation will be unavailable. The current design is misleading and might lead to mistakes.

Recommendation

The `cooldown` variable should represent the number of blocks for which liquidations will be unavailable. When setting `liquidation-cooldown-block`, the system should fetch the current `block-height` and add the `cooldown` variable to it.

```
// Change `cooldown` to be a diff, i.e., `block-height + cooldown`
(var-set liquidation-cooldown-block (+ block-height cooldown))
```

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
[H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
[M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
[M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
[M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
[M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
[L-01] Rounding Interest Against Protocol 15
[L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
[QA-01] Liquidation Flag Setter Can Be Improved 17
[QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
[QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
[QA-04] Market Asset Value Should Be Rounded Up 20
[QA-05] Simplify socialize-bad-debt 21

ClarityAlliance
Security Review
Granite

[QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged

Description

Currently, the system uses different yet duplicated functions when adding or removing collateral from the protocol. The functions `get-add-collateral-params` and `get-remove-collateral-params` can be merged into a single function called `get-collateral-params` .

Recommendation

The merged function should retain the initial logic from `get-add-collateral-params` , as any attempts to remove collateral with no position or insufficient collateral will result in a revert when executing the rest of the logic in the `remove-collateral` function.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
[H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
[M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
[M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
[M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
[M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
[L-01] Rounding Interest Against Protocol 15
[L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
[QA-01] Liquidation Flag Setter Can Be Improved 17
[QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
[QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
[QA-04] Market Asset Value Should Be Rounded Up 20
[QA-05] Simplify socialize-bad-debt 21

ClarityAlliance
Security Review
Granite

[QA-03] Usage of Additional Mapping `user-borrowed-block` is Not Necessarily Needed

Description

Currently, the mapping `user-borrowed-block` is used only once during the borrowing process. This mapping is not necessarily needed and can be removed. Instead, the positions struct could be updated to include an additional variable for the user's last borrowed block, which will be set when a user borrows.

Recommendation

Remove `user-borrowed-block` and update the positions struct with an additional variable to hold the user's last borrowed block.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
5.1. Impact 4
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
[H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
[M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
[M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
[M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
[M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
[L-01] Rounding Interest Against Protocol 15
[L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
[QA-01] Liquidation Flag Setter Can Be Improved 17
[QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
[QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
[QA-04] Market Asset Value Should Be Rounded Up 20
[QA-05] Simplify socialize-bad-debt 21

[QA-04] Market Asset Value Should Be Rounded Up

Description

Rounding down occurs when calculating the initial market value of an asset.

```
(ok (/ (* amount market-asset-price) scaling-factor))
```

Recommendation

Ensure that the function `get-market-asset-value` uses rounding up.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Granite 4
5. Risk Classification 4
 5.1. Impact 4
 5.2. Likelihood 5
 5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Findings 7
8.1. High Findings 9
 [H-01] [H-01] Not Calling accrue-interest Before a Safety Module Balance Mutation 9
8.2. Medium Findings 11
 [M-01] Missing Check to Ensure Removal of Collateral is Enabled When Withdrawing Full Collateral Position 11
 [M-02] Liquidation Doesn't Update the User's Collateral List if Their Collateral Position is Reduced to Zero 12
 [M-03] last-accrued-block Not Set When Mutating interest-accrual-enabled 13
 [M-04] Bad Debt Socialization Race Condition 14
8.3. Low Findings 15
 [L-01] Rounding Interest Against Protocol 15
 [L-02] Incorrect Interest Read-Only Output 16
8.4. QA Findings 17
 [QA-01] Liquidation Flag Setter Can Be Improved 17
 [QA-02] Duplicated Functions for Fetching Collateral Parameters Can Be Merged 18
 [QA-03] Usage of Additional Mapping user-borrowed-block is Not Necessarily Needed 19
 [QA-04] Market Asset Value Should Be Rounded Up 20
 [QA-05] Simplify socialize-bad-debt 21

ClarityAlliance
Security Review
Granite

[QA-05] Simplify `socialize-bad-debt`

Description

Rounding down occurs when calculating the initial market value of an asset.

```
(define-private (socialize-bad-debt (bad-debt bool) (user principal))
  (if (not bad-debt)
    SUCCESS
    ;; ...
```

Recommendation

Consider removing the `(if (not bad-debt))` nesting block and not passing `bad-debt` to `socialize-bad-debt` . Instead, call the function conditionally.