ClarityAlliance

ZEST PROTOCOL E-MODE SECURITY REVIEW

Conducted by:
KRISTIAN APOSTOLOV, ALIN BARBATEI (ABA), MARCHEV

NOVEMBER 15TH, 2024

CONTENTS

ClarityAlliance
Security Review

Zest Protocol
E-Mode

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

ClarityAlliance
Security Review

Zest Protocol
E-Mode

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 Zest Protocol's E-Mode implementation, where Clarity Alliance reviewed the scope, whilst simultaneously building out a testing suite for the protocol.

4. About Zest Protocol

Zest Protocol is a decentralized lending platform on Stacks that enables users to trustlessly lend and borrow assets.

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

ClarityAlliance
Security Review

Zest Protocol
E-Mode

CONTENTS

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

ClarityAlliance
Security Review

Zest Protocol E-Mode

6. Security Assessment Summary

Review Commit Hash:
54833b4ce5f9c2b9733a981d4ce8faa7a11daf90

7. Executive Summary

Over the course of the security review, Kristian Apostolov, Alin Barbatei (ABA), Marchev engaged with Zest Protocol to review Zest Protocol. In this period of time a total of **20** issues were uncovered.

Protocol Summary

Protocol Name	Zest Protocol
Repository	https://github.com/Zest-Protocol/zest-contracts
Date	December 16th, 2024
Protocol Type	Lending Protocol

Findings Count

Severity	Amount
Medium	4
Low	5
QA	11
Total Findings	**20**

CONTENTS

Summary of Findings

ID	Title	Severity	Status
[M-01]	Critical E-Mode Functionality Not Exposed Through pool-borrow-v2-0 to Configurator	Medium	Resolved
[M-02]	Lack of Staleness Checks for Oracle Price Data	Medium	Resolved
[M-03]	Assumption of aeUSDC Peg Can Lead to Manipulation and Undercollateralization Risks	Medium	Acknowledged
[M-04]	Asset Reserve Data Must Be Updated to Support Stacks Block Flow	Medium	Resolved
[L-01]	pool-reserve-data-2 does not approve the pool-borrow-v2-0 contract	Low	Resolved
[L-02]	Remove Testing Leftover	Low	Resolved
[L-03]	Duplicate Error Codes	Low	Resolved
[L-04]	Potential Loss of Ownership	Low	Resolved
[L-05]	Authorization via tx-sender is Vulnerable to Phishing Attacks	Low	Resolved
[QA-01]	Misaligned Function Name: set-asset-e-mode-types	QA	Resolved
[QA-02]	Remove Unused Constants	QA	Resolved
[QA-03]	Misleading Parameter Name in User E-Mode Functions	QA	Resolved
[QA-04]	Misleading Function Name: assets-are-of-e-mode-type	QA	Resolved
[QA-05]	Remove Unused Variable	QA	Resolved
[QA-06]	Inconsistent Naming in E-mode Configuration Functions	QA	Resolved
[QA-07]	Redundant LP Trait Parameter in Flashloan Logic	QA	Resolved
[QA-08]	Inconsistency in Verbosity Print Level in borrow-helper-v2-0	QA	Resolved
[QA-09]	Missing Error Codes	QA	Resolved
[QA-10]	Redundant SIP-10 Data Duplication in zstSTX v3	QA	Resolved
[QA-11]	Potential Forced Rounding Up When Compounding Borrowed Amount	QA	Acknowledged

CONTENTS

ClarityAlliance
Security Review

Zest Protocol
E-Mode

8. Findings

8.1. Medium Findings

[M-01] Critical E-Mode Functionality Not Exposed Through `pool-borrow-v2-0` to Configurator

Description

The `pool-borrow-v2-0` contract is designed to serve as a wrapper that invokes underlying functions from state contracts, such as `pool-reserve-data*`. However, certain functions from the `pool-reserve-data-2` contract are not wrapped in the `pool-borrow-v2-0` contract, unlike the `set-freeze-end-block` function. These functions include:

- `set-asset-e-mode-types`
- `set-asset-e-mode-type`
- `set-type-e-mode-config`

As a result, only direct calls to the `pool-reserve-data-2` contract can modify these configurations, which is not the intended behavior. The `pool-reserve-data-2` contract is designed to check for approved contracts, not the callers themselves.

Recommendation

Implement wrappers in the `pool-borrow-v2-0` contract for the `set-asset-e-mode-types`, `set-asset-e-mode-type`, and `set-type-e-mode-config` functions, making them callable by the configurator role.

CONTENTS

[M-02] Lack of Staleness Checks for Oracle Price Data

Description

The protocol depends on the Arkadiko oracle for asset price data, which includes a `last-block` field indicating the block height of the most recent data update.

Currently, the oracle adapters in the project retrieve only the latest price data without checking its freshness. For example:

```
(define-public (get-asset-price (token <ft>))
  (let (
    (oracle-data
      (contract-call? 'SP2C2YFP12AJZB4MABJBAJ55XECVS7E4PMMZ89YZR.arkadiko-oracle-v2-3
      get-price
      "STX"
    ))
  )
    ;; convert to fixed precision
    (ok (to-fixed (get last-price oracle-data) u6))
  )
)
```

If the oracle fails to provide timely updates, this implementation may result in the protocol using outdated price data. This could pose risks to various functionalities of the protocol, as the oracle price data influences key operations such as reserve management and liquidation decisions.

Recommendation

Implement a staleness check to ensure the freshness of oracle data. The protocol should only accept price data if it has been updated within a specified number of recent blocks (e.g., within `x` blocks).

CONTENTS

[M-03] Assumption of aeUSDC Peg Can Lead to Manipulation and Undercollateralization Risks

Description

The protocol currently assumes that the value of aeUSDC is pegged at $1:

```
(define-read-only (get-price)
  (let (
    (oracle-data
      (contract-call? 'SP2C2YFP12AJZB4MABJBAJ55XECVS7E4PMMZ89YZR.arkadiko-oracle-v2-3
      get-price
      "STX"
    ))
  )
    ;; convert to fixed precision
    u100000000 ;; @audit Hardcoded value of $1
  )
)
```

However, aeUSDC does not maintain a strict peg and can fluctuate above or below $1, as demonstrated by its price variations on CoinGecko. This assumption poses a risk because if the price of aeUSDC drops below $1 or the token significantly depegs, the protocol will continue to value aeUSDC collateral at a higher rate than its actual market price.

Consequently, borrowers using aeUSDC as collateral may avoid liquidation even when undercollateralized, creating a vulnerability in the liquidation logic and potentially leading to the accumulation of bad debt within the protocol.

Conversely, if aeUSDC's market value exceeds $1, the hard peg at $1 will result in overcollateralization and could trigger liquidations that would not occur if aeUSDC were priced accurately.

Recommendation

Consider integrating a price oracle to obtain the real-time market price of aeUSDC instead of assuming a fixed $1 value. This approach would ensure that collateral is priced accurately, reflecting the true market value of aeUSDC, and help maintain the protocol's security and financial stability.

CONTENTS

[M-04] Asset Reserve Data Must Be Updated to Support Stacks Block Flow

Description

Each asset in the protocol has associated reserve data information, which is used in several critical areas.

With the introduction of the new v3 contracts, the protocol has transitioned to using the Stacks chain block height (`stacks-block-height`) instead of the Bitcoin chain block height (`burn-block-height`) for timekeeping.

To implement this change correctly, certain configurations must be modified; otherwise, several severe issues may arise.

The data that needs to be updated includes:

- `current-liquidity-rate` and `last-liquidity-cumulative-index` : Incorrect rates may be applied if these are not updated.
- `last-updated-block` : When determining the compounded borrow balance, the difference between the latest recorded block and the current block is calculated as `(- stacks-block-height last-updated-block)` .

While this operation will succeed after an internal accounting update that changes the latest recorded block to a Stacks block number, until that update is completed, operations are blocked due to an underflow.

The underflow occurs because the `last-updated-block` saved before the deployment of v3 relied on the Bitcoin block height, which is in the 870,000 range, whereas the Stacks block height is in the 200,000 range.

Recommendation

When updating from v2 to v3, call `pool-0-reserve-v2-0::set-reserve` to change the `last-updated-block` to the current Stacks block height. Also, use appropriate values for the other configurations.

CONTENTS

ClarityAlliance
Security Review

Zest Protocol
E-Mode

8.2. Low Findings

[L-01] pool-reserve-data-2 does not approve the pool-borrow-v2-0 contract

Description

Upon deployment, the `pool-reserve-data-2` contract mistakenly approves the `pool-borrow-v1-2` contract instead of the intended `pool-borrow-v2-0` contract.

```
(map-set approved-contracts .pool-borrow-v1-2 true)
```

The deployment system anticipates a pattern similar to how `pool-reserve-data-1` approves `.pool-borrow-v1-2` or how `pool-reserve-data` approves `.pool-borrow` and `.pool-0-reserve`.

As a result, initial operations through the `pool-borrow-v2-0` contract would fail, requiring administrative intervention to manually add the contract using the `pool-reserve-data-2::set-approved-contract` call.

Recommendation

Modify line 87 in `pool-reserve-data-2.clar` to replace `pool-borrow-v1-2` with `pool-borrow-v2-0`.

CONTENTS

[L-02] Remove Testing Leftover

Description

The `math-v2-0.clar` contract includes the following remnant from testing:

```
(stx-transfer? u100000000 tx-sender 'STC6G8DC2A0V58A6399M22C06BF4EK5JZSQW7BWP)
```

Recommendation

Eliminate the testing leftover implementation mentioned above.

ClarityAlliance
Security Review

Zest Protocol
E-Mode

CONTENTS

[L-03] Duplicate Error Codes

Description

The `pool-0-reserve-v2-0` contract introduces the following constants representing error codes:

```
(define-constant ERR_INVALID_VALUE (err u7005))
(define-constant ERR_E_MODE_DOES_NOT_EXIST (err u7006))
(define-constant ERR_CANNOT_BORROW_DIFFERENT_E_MODE_TYPE (err u7007))
```

However, these error codes are already in use by other constants:

```
(define-constant ERR_NON_ZERO (err u7005))
(define-constant ERR_OPTIMAL_UTILIZATION_RATE_NOT_SET (err u7006))
(define-constant ERR_BASE_VARIABLE_BORROW_RATE_NOT_SET (err u7007))
```

This duplication of error codes can lead to confusion and issues during error rendering or debugging.

Recommendation

Assign a new set of error codes for any new errors in the contract:

```
@@ -33,9 +33,9 @@
  (define-constant ERR_HEALTH_FACTOR_LIQUIDATION_THRESHOLD (err u7010))
  (define-constant ERR_FLASHLOAN_FEE_TOTAL_NOT_SET (err u7011))
  (define_constant ERR_FLASHLOAN_FEE_PROTOCOL_NOT_SET (err u7012))
-(define-constant ERR_INVALID_VALUE (err u7005))
-(define-constant ERR_E_MODE_DOES_NOT_EXIST (err u7006))
-(define-constant ERR_CANNOT_BORROW_DIFFERENT_E_MODE_TYPE (err u7007))
+(define-constant ERR_INVALID_VALUE (err u7013))
+(define-constant ERR_E_MODE_DOES_NOT_EXIST (err u7014))
+(define-constant ERR_CANNOT_BORROW_DIFFERENT_E_MODE_TYPE (err u7015))

  (define-public (set-flashloan-fee-total (asset principal) (fee uint))
    (begin
```

CONTENTS

[L-04] Potential Loss of Ownership

Description

In the `pool-reserve-data-2`, `pool-borrow-v2-0`, and `lp-ststx-v3` contracts, ownership is updated through the `set-contract-owner` or `set-configurator` functions.

However, the new owner being set is not validated using the `is-standard` function. This lack of validation could lead to potential misconfigurations where an invalid principal is assigned ownership.

Furthermore, the ownership transfer occurs in a single step, which poses a risk to the protocol's security and maintenance if an incorrect or inaccessible principal is configured as the new owner.

Recommendation

Implement a two-step ownership transfer mechanism to enhance the security of the ownership transfer:

1. The current owner initiates the ownership transfer by proposing a new owner, marking the transfer as pending.
2. The new principal must then explicitly accept the ownership transfer for it to take effect.

Additionally, validate the new owner principal with `is-standard` to ensure it is valid on the current network.

CONTENTS

[L-05] Authorization via tx-sender is Vulnerable to Phishing Attacks

Description

The `pool-borrow-v2-0` contract uses `tx-sender` to authorize several sensitive operations, such as `supply`, `withdraw`, `borrow`, and `repay`.

Although `tx-sender` identifies the sender of the transaction, it does not verify the context from which the call is made. This vulnerability exposes users to phishing attacks, where a malicious contract can deceive a user into executing unintended transactions.

In these scenarios, users might unknowingly perform sensitive operations on the `pool-borrow-v2-0` contract, which could lead to unauthorized asset transfers.

Recommendation

Implement `contract-caller` instead of `tx-sender` as an authorization mechanism to mitigate the risk of phishing attacks.

CONTENTS

8.3. QA Findings

[QA-01] Misaligned Function Name: set-asset-e-mode-types

Description

The function `pool-reserve-data-2::set-asset-e-mode-types` does not adhere to the naming convention used across all `pool-reserve-data*` contracts. This inconsistency is apparent from both the section comment and the subsequent getter names.

This misalignment reduces the readability and understanding of the code.

Recommendation

Rename `set-asset-e-mode-types` to `set-e-mode-types`.

CONTENTS

ClarityAlliance
Security Review

Zest Protocol
E-Mode

[QA-02] Remove Unused Constants

Description

The `math-v2-0.clar` contract includes several constants that are not being utilized:

- `one-12`
- `seconds-in-block` (employed in a getter function that defines an unused read-only function in `pool-0-reserve-read` and `pool-0-reserve-v2-0`)
- `get-sb-by-sy` (employed in a getter function that defines an unused read-only function in `pool-0-reserve-v2-0`)

Recommendation

Eliminate any unused constants to enhance the readability and maintainability of the codebase.

CONTENTS

[QA-03] Misleading Parameter Name in User E-Mode Functions

Description

In the `pool-reserve-data-2` contract, all functions related to user e-mode use a parameter named `asset`, which is misleading.

In these functions— `set-user-e-mode`, `set-user-e-mode`, and `get-user-e-mode-read` — the `asset` parameter actually represents the user principal for which the e-mode configuration is set or queried, not an asset.

This inconsistency could lead to confusion and diminish code readability and maintainability, as the parameter name does not accurately describe its purpose.

Recommendation

Rename the `asset` parameter in these functions to `user`, reflecting its actual purpose in the code:

CONTENTS

[QA-04] Misleading Function Name: `assets-are-of-e-mode-type`

Description

The function name `assets-are-of-e-mode-type` implies that it checks the E-mode type for multiple assets. However, the current implementation only supports verifying the E-mode type for a single asset, not a list.

This discrepancy between the function name and its actual behavior can cause confusion and diminish code readability.

Recommendation

Rename the function to more accurately reflect its purpose, such as `asset-is-of-e-mode-type`.

CONTENTS

[QA-05] Remove Unused Variable

Description

Within the function, the variable `collateral-balance-in-base-currency` is declared to store the user's total collateral balance in base currency. However, it is not consistently utilized throughout the function.

For example, in the calculation of `collateral-balance-after-decrease`, the function directly uses `(get total-collateral-balanceUSD user-global-data)` instead of the declared `collateral-balance-in-base-currency` variable. This inconsistency reduces code readability by introducing redundancy and undermining the purpose of the defined variable.

Recommendation

Ensure the `collateral-balance-in-base-currency` variable is used consistently in all relevant parts of the function to enhance consistency and readability.

CONTENTS

[QA-06] Inconsistent Naming in E-mode Configuration Functions

Description

In the E-mode type configuration functions, the naming convention employs the term `type-e-mode`, as observed in functions such as `set-type-e-mode-config`, `get-type-e-mode-config`, and `get-type-e-mode-config-read`.

Conversely, in other parts of the codebase, such as user, asset, and E-mode type management functions, the term `e-mode-type` is consistently used to refer to the E-mode type. This inconsistency in naming conventions can diminish code readability and maintainability.

Recommendation

Standardize the naming convention across all E-mode-related functions to consistently use either `e-mode-type` or `type-e-mode`.

CONTENTS

[QA-07] Redundant LP Trait Parameter in Flashloan Logic

Description

The flashloan functionality, beginning with `borrow-helper-v2-0::flashloan` and continuing in `pool-borrow-v2-0::flashloan`, requires the `lp <ft>` principal trait to be passed. This trait is validated to ensure it is the correct `a-token` contract. However, the LP token serves no other purpose and does not contribute additional functionality, making its inclusion unnecessary.

Recommendation

Remove the redundant LP trait argument from the flashloan functionality.

CONTENTS

ClarityAlliance
Security Review

Zest Protocol
E-Mode

[QA-08] Inconsistency in Verbosity Print Level in borrow-helper-v2-0

Description

In `borrow-helper-*` type contracts, print statements are included for all major operations, such as liquidations and withdrawals.

Additionally, the `flashloan` functions include print statements within the `pool-borrow-v2-0` contract.

However, the `borrow-helper-v2-0::set-e-mode` function lacks print statements in both the `borrow-helper-v2-0` and `pool-borrow-v2-0` contracts, resulting in inconsistent logging for this specific function.

Recommendation

Incorporate print statements into the `borrow-helper-v2-0::set-e-mode` function.

CONTENTS

[QA-09] Missing Error Codes

Description

Each contract in the codebase is assigned a distinct range of error codes to facilitate easier debugging.

In `pool-borrow-v2-0` contract, the error code `u30016`, which was associated with the now-removed `ERR_REPAYMENT_SHOULD_BE_EXACT` error, is absent, resulting in a gap in the error codes.

```
(define-constant ERR_FLASHLOAN_DISABLED (err u30015))
(define-constant ERR_REPAY_BEFORE_DISABLING (err u30017))
```

Recommendation

To enhance code readability and simplify third-party integration with the contract in the event of errors, assign the last error code (`ERR_MUST_BORROW_E_MODE_TYPE`) to fill the missing gap.

CONTENTS

[QA-10] Redundant SIP-10 Data Duplication in zstSTX v3

Description

The latest version of `lp-ststx-v3`, which builds upon the `lp-ststx-v2` implementation with timekeeping modifications, forwards all SIP-10 calls such as `get-total-supply`, `get-name`, `get-decimals`, and `get-token-uri`.

However, `lp-ststx-v3` still contains three local data variables with default values that are not utilized:

```
(define-data-var token-uri
//(string-utf8 256) u"<https://token-meta.s3.eu-central-1.amazonaws.com/zstSTX.json>")
(define-data-var token-name (string-ascii 32) "Zest stSTX")
(define-data-var token-symbol (string-ascii 32) "zstSTX")
```

These variables are redundant and will result in a slight increase in fees when loading the zstSTX contract.

Recommendation

If these variables are meant to aid code readability, consider leaving only the values as comments. Otherwise, remove the three data variables.

CONTENTS

[QA-11] Potential Forced Rounding Up When Compounding Borrowed Amount

Description

The calculation of a user's compounded borrowed amount with interest is performed using the `pool-0-reserve-v2-0::get-compounded-borrow-balance` function.

To address potential rounding down, an additional unit is added to the borrowing amount if the user's reserve data was updated in a block different from the current one and if the compounded borrowing balance equals the entire borrowed amount.

```
(compounded-balance
  (mul-precision-with-factor principal-borrow-balance decimals cumulated-interest)))
(if (is-eq compounded-balance principal-borrow-balance)
  ;; Add 1 in case of rounding down
  (if (not (is-eq last-updated-block stacks-block-height))
    (+ principal-borrow-balance u1)
    compounded-balance
    )
    compounded-balance
)
```

After transitioning to using Stacks block height for timekeeping, the `last-updated-block` variable will reflect a Bitcoin block height, while the new `stacks-block-height` command will return the Stacks block height, leading to an automatic rounding up.

Migrating contracts from v2 to v3 will involve pausing the initial v2 system before proceeding with the v3 upgrades. Consequently, this condition may also occur naturally.

Recommendation

Recognize this issue as a forced rounding of one unit. Given the unlikely occurrence of the required preconditions, further contract changes are not justified.