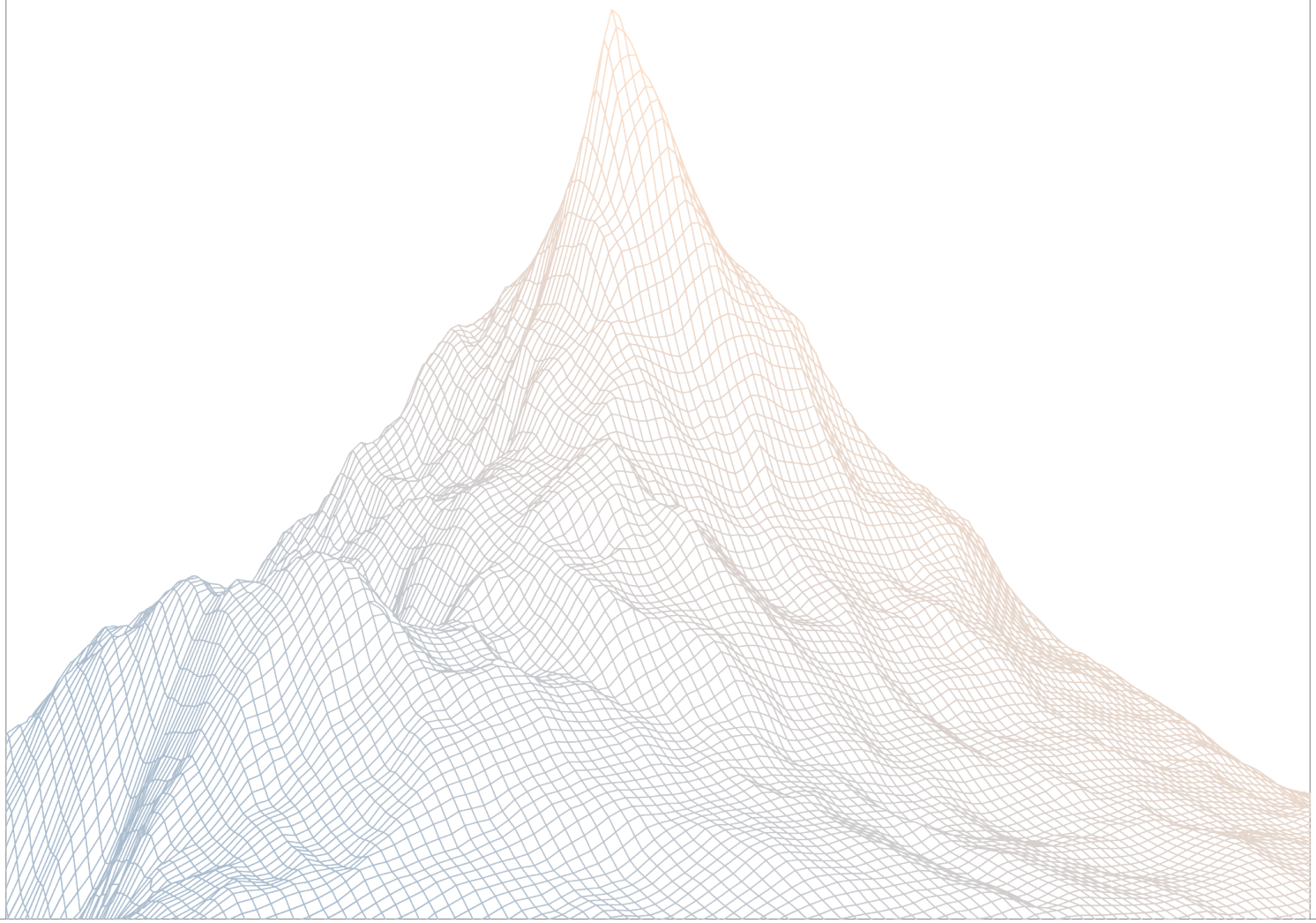
Zenith

Vicuna Finance

Smart Contract Security Assessment

VERSION 1.1



AUDIT DATES: February 28th to February 28th, 2025

AUDITED BY: defsec
said

Contents

1 Introduction

1.1 About Zenith

Zenith is an offering by Code4rena that provides consultative audits from the very best security researchers in the space. We focus on crafting a tailored security team specifically for the needs of your codebase.

Learn more about us at https://code4rena.com/zenith.

1.2 Disclaimer

This report reflects an analysis conducted within a defined scope and time frame, based on provided materials and documentation. It does not encompass all possible vulnerabilities and should not be considered exhaustive.

The review and accompanying report are presented on an "as-is" and "as-available" basis, without any express or implied warranties.

Furthermore, this report neither endorses any specific project or team nor assures the complete security of the project.

1.3 Risk Classification

SEVERITY LEVEL	IMPACT: HIGH	IMPACT: MEDIUM	IMPACT: LOW
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

2

Executive Summary

2.1 About Vicuna Finance

Vicuna Finance operates as a native borrow/lending protocol, facilitating leveraged yield farming on the Sonic Blockchain. Lenders can earn yields on their tokens, while borrowers gain access to capital-efficient, undercollateralized loans for leveraged yield farming positions. The platform leverages integrated exchanges' liquidity layers, linking LP borrowers and lenders to enhance capital efficiency within the ecosystem.

2.2 Scope

The engagement involved a review of the following targets:

Target	Vicuna-core-bl
Repository	https://github.com/VicunaFinance-com/Vicuna-core-bl
Commit Hash	`782f51917056a53a2c228701058a6c3fb233684a`
Files	`src/*` `contracts/*`

2.3 Audit Timeline

February 28, 2025	Audit start
February 28, 2025	Audit end
March 14, 2025	Report published

2.4 Issues Found

SEVERITY	COUNT
Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	5
Informational	3
Total Issues	**8**

3

Findings Summary

ID	Description	Status
L-1	Misleading token name implementation	Acknowledged
L-2	Missing state and interest rate updates in setReserveInterestRateStrategyAddress	Acknowledged
L-3	Implementation of consistent methodology for setting supply and borrow caps	Acknowledged
L-4	API3 and Pegged Oracles revert on invalid prices, potentially blocking the fallback oracle if configured	Acknowledged
L-5	Lack of initial deposit in the new market listing script	Acknowledged
I-1	Upgrade privileged roles to multi-sig after the deployment	Acknowledged
I-2	Unverified contracts on SonicScan	Resolved
I-3	Pyth Price Oracle might break some assumptions	Acknowledged

4

Findings

4.1 Low Risk

A total of 5 low risk findings were identified.

[L-1] Misleading token name implementation

SEVERITY: Low	IMPACT: Low
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- 0xF224CB039F2B5909197c019b1972E62d7fdCdA0f

Description:

The ERC20 token contract at address `0xF224CB039F2B5909197c019b1972E62d7fdCdA0f` has implemented a problematic token name configuration. Instead of using a descriptive name that identifies the token's purpose or brand (like "Vicuna Sonic USDC.E"), the token name is simply set to "ERC20".

Users will see "ERC20" as the token name in wallets and explorers, which is the name of the token standard itself, not a unique identifier.

Recommendations:

Use "Vicuna Sonic USDC.E" or a similar descriptive name that accurately represents the token's purpose.

Vicuna: Acknowledged.

[L-2] Missing state and interest rate updates in `setReserveInterestRateStrategyAddress`

SEVERITY: Low	IMPACT: Low
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- Pool.sol#L628

Description:

In the the `setReserveInterestRateStrategyAddress` function are not updating the reserve's state and interest rates when the interest rate strategy address is changed. Without these updates, the reserve's state and interest rates may become inconsistent, leading to incorrect calculations or unexpected behavior.

```
/// @inheritdoc IPool
function setReserveInterestRateStrategyAddress(
  address asset,
  address rateStrategyAddress
) external virtual override onlyPoolConfigurator {
  require(asset ≠ address(0), Errors.ZERO_ADDRESS_NOT_VALID);
  require(_reserves[asset].id ≠ 0 || _reservesList[0] == asset,
    Errors.ASSET_NOT_LISTED);
  _reserves[asset].interestRateStrategyAddress = rateStrategyAddress;
}
```

Recommendations:

Introduce the calls to `reserve.updateState` and `reserve.updateInterestRates` to ensure that the reserve's state and interest rates are updated correctly when the interest rate strategy address is changed.

```
/// @inheritdoc IPool
function setReserveInterestRateStrategyAddress(
  address asset,
  address rateStrategyAddress
) external virtual override onlyPoolConfigurator {
```

```
    require(asset ≠ address(0), Errors.ZERO_ADDRESS_NOT_VALID);
    require(_reserves[asset].id ≠ 0 || _reservesList[0] == asset,
      Errors.ASSET_NOT_LISTED);

    DataTypes.ReserveData storage reserve = _reserves[asset];
    DataTypes.ReserveCache memory reserveCache = reserve.cache();

    reserve.updateState(reserveCache);
    _reserves[asset].interestRateStrategyAddress = rateStrategyAddress;
    reserve.updateInterestRates(reserveCache, asset, 0, 0);
  }
```

Vicuna: Acknowledged

[L-3] Implementation of consistent methodology for setting supply and borrow caps

SEVERITY: Low	IMPACT: Low
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- 0x4e1dee6eb2f48bcf26a7dca815b4b91e149bf21c791d2dbc8b5fc42e67277599

Description:

In the market parameters, supply and borrow caps are critical risk management parameters that help maintain the protocol's stability and security. These caps are designed to limit the amount of an asset that can be supplied or borrowed, thereby mitigating risks such as price exploits and protocol insolvency.

However, many assets currently have their supply and borrow caps set to "0", indicating that these parameters have not been fully implemented or optimized. This situation necessitates the development of a transparent and consistent methodology for setting these caps, especially in light of recent market volatility and changes in sentiment.

The current state of supply and borrow caps in the Vicuna is inconsistent, with many assets lacking defined caps. The absence of a standardized methodology for setting these caps further complicates the issue, as it leaves the protocol exposed to risks associated with market fluctuations.

Recommendations:

Adopt the outlined methodology for setting supply and borrow caps across all assets in the deployment.

Vicuna: scETH is currently set to 0 because the oracle doesn't work as expected. We will correctly set it when the oracle is fixed

Zenith: Acknowledged

[L-4] API3 and Pegged Oracles revert on invalid prices, potentially blocking the fallback oracle if configured

SEVERITY: Low	IMPACT: Low
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- Oracles

Description:

Here is the current implementation inside `PeggedOracle`, used by `sTS` and `wOS`.

```
  function getChainlinkPrice() public view returns (int256) {
    IChainlinkOracle chainlinkOracle = IChainlinkOracle(peggedAssetSource);
      uint256 _decimals = chainlinkOracle.decimals();
      (, int256 price, , , ) = chainlinkOracle.latestRoundData();
      if (_decimals > STANDARD_DECIMALS) {
          price =  price / int256((10 ** (_decimals - STANDARD_DECIMALS)));
      } else if (_decimals < STANDARD_DECIMALS) {
          price = price * int256((10 ** (STANDARD_DECIMALS - _decimals)));
      }
      if (price > 0) {
          return price;
      }
>>>   revert("Chainlink : price not found");
  }

  /**
   * @notice Gets the price from the Pyth oracle
   * @return The normalized price with 8 decimals
   */
  function getPythPrice() public view returns (int256) {
    IPyth pythOracle = IPyth(peggedAssetSource);
    Price memory priceData = pythOracle.getPrice(pythPriceId);
    int32 expo = priceData.expo;
    int64 price = priceData.price;
>>> require(price > 0, "Pyth response : price is negative or null");
    // uint64 unsignedPrice = uint64(price);
    if (expo > TARGET_EXPO) {
```

```
            return price / int256((10 ** uint32(expo - TARGET_EXPO)));
        } else if (expo < TARGET_EXPO) {
            return price * int256((10 ** uint32(TARGET_EXPO - expo)));
        } else {
            return price;
        }
    }

    /**
     * @notice Gets the price from API3 Oracles
     * @return The normalized price with 8 decimals
     */
    function getAPI3Price() public view returns (int256) {
      AggregatorInterface source = AggregatorInterface(peggedAssetSource);
      int256 price = source.latestAnswer();
      uint8 _decimals = source.decimals();
      if (_decimals < STANDARD_DECIMALS) {
          price = price * int256(10**(STANDARD_DECIMALS - _decimals));
      }
      if (_decimals > STANDARD_DECIMALS) {
          price = price / int256(10**(_decimals - STANDARD_DECIMALS));
      }
      if (price > 0) {
          return price;
      }
>>> revert("API3 : price not found");
    }
```

And here is the API3 oracle implementation for getting price :

```
    function read()
        public
        view
        override
        returns (int224 value, uint32 timestamp)
    {
        bytes32 dataFeedId = IApi3ServerV1(api3ServerV1)
            .dapiNameHashToDataFeedId(dapiNameHash);
        if (dataFeedId == bytes32(0)) {
            revert DapiNameIsNotSet();
        }
        (int224 baseDapiValue, uint32 baseDapiTimestamp) = IApi3ServerV1(
            api3ServerV1
        ).dataFeeds(dataFeedId);
        (
            int224 oevDapiValue,
```

```
            uint32 oevDapiTimestamp
        ) = IApi3ServerV1OevExtension(api3ServerV1OevExtension).oevDataFeed(
                dappId,
                dataFeedId
            );
        if (oevDapiTimestamp > baseDapiTimestamp) {
            (value, timestamp) = (oevDapiValue, oevDapiTimestamp);
        } else {
            (value, timestamp) = (baseDapiValue, baseDapiTimestamp);
        }
        if (timestamp == 0) {
>>>         revert DataFeedIsNotInitialized();
        }
    }
```

In these implementations, the oracles revert in the case of an invalid price. However, in Aave, a fallback oracle can be configured to retrieve the price if the main oracle returns an invalid value.

```
  function getAssetPrice(address asset)
    public view override returns (uint256) {
    AggregatorInterface source = assetsSources[asset];

    if (asset == BASE_CURRENCY) {
      return BASE_CURRENCY_UNIT;
    } else if (address(source) == address(0)) {
      return _fallbackOracle.getAssetPrice(asset);
    } else {
      int256 price = source.latestAnswer();
      if (price > 0) {
        return uint256(price);
      } else {
>>>     return _fallbackOracle.getAssetPrice(asset);
      }
    }
  }
```

If the main price feed reverts on an invalid price, the fallback oracle will never function if configured.

Recommendations:

Consider returning a price of 0 when the price feed returns invalid data.

Vicuna: Acknowledged

[L-5] Lack of initial deposit in the new market listing script

SEVERITY: Low	IMPACT: Low
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- deploy/02_market

Description:

The new market listing script does not include or bundle a small initial deposit. This could make the new market prone to the known empty market exploit.

Recommendations:

When listing new market, consider always bundling the listing with a small initial deposit to avoid the empty market issue. For reference, this is the Aave's deployment transaction : here.

Vicuna: Acknowledged. Our new script will be deployed from our contract multisig and will have a initial bundled deposit on the script to prevent "empty market exploit"

4.2 Informational

A total of 3 informational findings were identified.

[I-1] Upgrade privileged roles to multi-sig after the deployment

SEVERITY: Informational	IMPACT: Informational
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- 0xbE0B2230B842be6A37188038a58755534dC9E999

Description:

In the deployment, the market owner, emergency admin, and pool admin roles are currently assigned to a single deployer address.

This configuration centralizes control, increasing the risk of unauthorized access or misuse. By transitioning these roles to a multi-sig wallet, the protocol can distribute control among multiple parties, reducing the risk of single-point failures.

Recommendations:

Transition the market owner, emergency admin, and pool admin roles to multi-sig wallets. This will require multiple approvals for critical actions.

Vicuna: Acknowledged

[I-2] Unverified contracts on SonicScan

SEVERITY: Informational	IMPACT: Informational
STATUS: Resolved	LIKELIHOOD: N/A

Target

- 0xDdE41Eb19346D423b4Cc8e3917c3000cFBE490Fa, 0xc9d2f7fd8f7025a9696d1ec9a5b2318602c431df

Description:

The contract at address `0xDdE41Eb19346D423b4Cc8e3917c3000cFBE490Fa` on SonicScan is currently **unverified**. There are also other contracts which are not verified.

Recommendations:

The contract owner should verify the contract by uploading the source code to SonicScan. This will make the contract's functionality transparent and allow for public scrutiny.

Vicuna: Verified on-chain through explorer: 0xDdE41Eb19346D423b4Cc8e3917c3000cFBE490Fa, 0xc9d2f7fd8f7025a9696d1ec9a5b2318602c431df

Zenith: Verified

[I-3] Pyth Price Oracle might break some assumptions

SEVERITY: Informational	IMPACT: Informational
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- PythOracle

Description:

Ptyh is a pull oracle where anyone can permissionlessly update the on-chain price. It is important to note that since the Pyth oracle's price updates are unrestricted (with the only condition being that the timestamp in the payload is higher), a user can make this oracle return two different prices in the same transaction.

This means, for instance, a user can open a borrow position, then price is updated, causing it to become liquidatable and get liquidated within the same transaction.

Recommendations:

It is important to evaluate all potential risks associated with this Pyth oracle behavior and determine whether they are acceptable.

Vicuna: This point is acknowledged and we will dive into it further. Also contact the PYTH team