Treehouse tAVAX Audit Report

Aug 22, 2025

Table of Contents

Summary 2

Overview 3

Issues 4

[WP-M1] Current implementation cannot claim rewards from Aave's `Merit Programs` for asAVAX 4

[WP-L2] Approving the sAVAX contract to transfer tokens from this contract to itself during `requestUnlock` is unnecessary 5

[WP-L3] `NavErc20.nav()` Unexpectedly Reverts When `_nav` is Small 7

[WP-N4] The NatSpec documentation for `ASAVAXRateProvider` appears to be mistakenly copied from WAVAX documentation. 9

[WP-L5] Unnecessary precision loss and gas waste in `TAvaxExchangeRateProvider.getPooledAvaxByShares()` 10

[WP-M6] ChainlinkRateProvider fails to account for token decimals and updatedAt (whether the price data is stale) 12

Appendix 16

Disclaimer 17

Summary

This report has been prepared for Treehouse smart contract, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Overview

Project Summary

Project Name	**Treehouse**
Codebase	**https://github.com/treehouse-gaia/tAVAX-protocol**
Commit	**4b9a9d0881ded6b966172c5f587041bd50a349b8**
Language	**Solidity**

Audit Summary

Delivery Date	**Aug 22, 2025**
Audit Methodology	**Static Analysis, Manual Review**
Total Isssues	**6**

[WP-M1] Current implementation cannot claim rewards from Aave's `Merit Programs` for asAVAX

Medium

Issue Description

Aave is currently offering rewards for asAVAX through its `Merit Programs` . The reward APR is 0.32% at the time of writing.

To receive `asAVAX` rewards, the contract needs to implement `Merit Programs` reward claiming in the action.

The current implementation only claims internal Aave rewards from the `RewardsController` but not the rewards from the Merit Programs.

> Merit Programs Third-party programs that provide extra APR rewards for specific lending activities. These require separate claiming through external platforms.

> Aave Governance Rewards Incentives set by Aave governance through the RewardsController, typically distributed in AAVE tokens or other approved reward tokens.

Status

ⓘ Acknowledged

[WP-L2] Approving the sAVAX contract to transfer tokens from this contract to itself during `requestUnlock` is unnecessary

Low

Issue Description

https://github.com/treehouse-gaia/tAVAX-protocol/blob/1ad0b0d012dd2a543bd7f67717d055a8fd3df2c0/contracts/strategy/actions/savax/SavaxUnlock.sol#L41-L54

```
41  function _savaxUnlock(uint _amount) internal returns (uint amountUnlock, bytes
    memory logData) {
42      ISAVAX savax = ISAVAX(sAVAX);
43      amountUnlock = _amount;
44
45      // Check if we have enough sAVAX balance
46      uint balance = savax.balanceOf(address(this));
47      require(balance >= amountUnlock, 'Insufficient sAVAX balance');
48
49      // Approve sAVAX contract to transfer tokens from this contract to itself
    during requestUnlock
50      TokenUtils.approveToken(sAVAX, sAVAX, amountUnlock);
51      savax.requestUnlock(amountUnlock);
52
53      logData = abi.encode(sAVAX, amountUnlock);
54    }
```

```
222 /**
223     * @notice Start unlocking cooldown period for `shareAmount` AVAX
224     * @param shareAmount Amount of shares to unlock
225     */
226 function requestUnlock(uint shareAmount) external nonReentrant whenNotPaused {
227     require(shareAmount > 0, "Invalid unlock amount");
228     require(shareAmount <= shares[msg.sender], "Unlock amount too large");
229
230     userSharesInCustody[msg.sender] =
    userSharesInCustody[msg.sender].add(shareAmount);
231     _transfer(msg.sender, address(this), shareAmount);
```

```
232
233         userUnlockRequests[msg.sender].push(UnlockRequest(
234             block.timestamp,
235             shareAmount
236         ));
237
238         emit UnlockRequested(msg.sender, shareAmount);
239     }
```

Status

✓ Fixed

[WP-L3] `NavErc20.nav()` Unexpectedly Reverts When `_nav` is Small

Low

Issue Description

When `_nav` in `NavErc20.nav()` is so small that $\left\lfloor \frac{nav \times sAVAX.totalShares}{sAVAX.totalPooledAvax} \right\rfloor$ equals 0, `NavErc20.nav()` will unexpectedly revert.

For example, if the `_target` 's `_nav` is initially `0` , and someone transfers 1 wei of native token AVAX to `_target` before `NavErc20.nav()` executes, `NavErc20.nav()` L56 `sAVAX.getSharesByPooledAvax(_nav)` will unexpectedly revert due to the implementation in `sAVAX` L206.

The threshold of `_nav` that triggers the `Error("Invalid share count")` depends on how small the ratio $\frac{sAVAX.totalShares}{sAVAX.totalPooledAvax}$ is.

Currently, with `sAVAX.totalShares` at 1290036588203052760302088 2 and `sAVAX.totalPooledAvax` at 1575396380343469428351012 0, this issue only occurs when `_nav` equals 1.

`NavErc20WithDebt` has a similar issue.

```
25    /**
26     * @notice Calculate total NAV of native AVAX + ERC20 tokens for a target
    address
27     * @param _target Address to calculate NAV for
28     * @param _tokens Array of ERC20 token addresses
29     * @return _nav Total NAV in sAVAX terms
30     * @dev Returns raw sAVAX balance if no other assets, otherwise converts AVAX to
    sAVAX shares + raw sAVAX
31     * @dev Silently ignores tokens with zero rates from the registry
32     */
33    function nav(address _target, address[] memory _tokens) external view returns
    (uint _nav) {
34        _nav += _target.balance;
35
36        uint wip;
37        uint sAVAXBalance;
38        for (uint i; i < _tokens.length; ++i) {
```

```
39          wip = IERC20(_tokens[i]).balanceOf(_target);
40
41          if (wip > 0) {
42            if (_tokens[i] == address(sAVAX)) {
43              sAVAXBalance = wip;
44            } else {
45              uint rate = RATE_PROVIDER_REGISTRY.getRateInAvax(_tokens[i]);
46              if (rate > 0) {
47                _nav += Math.mulDiv(rate, wip, 1e18);
48              }
49              // Note: Zero-rate tokens are silently ignored
50            }
51          }
52        }
53        if (_nav == 0) {
54          _nav = sAVAXBalance;
55        } else {
56          _nav = sAVAX.getSharesByPooledAvax(_nav) + sAVAXBalance;
57        }
58      }
```

```
197       /**
198        * @return The amount of shares that corresponds to `avaxAmount`
      protocol-controlled AVAX.
199        */
200       function getSharesByPooledAvax(uint avaxAmount) public view returns (uint) {
201           if (totalPooledAvax == 0) {
202               return 0;
203           }
204
205           uint shares = avaxAmount.mul(totalShares).div(totalPooledAvax);
206           require(shares > 0, "Invalid share count");
207
208           return shares;
209       }
```

Status

✓ Fixed

[WP-N4] The NatSpec documentation for `ASAVAXRateProvider` appears to be mistakenly copied from WAVAX documentation.

Issue Description

The implementation is not a ”Wrapped avax Rate Provider”, and there is no ”hardcoded value of 1e18”.

Also, `sAVAX` on L32 might be a copy-paste error, consider changing `sAVAX` to `asAVAX` .

https://github.com/treehouse-gaia/tAVAX-protocol/blob/1ad0b0d012dd2a543bd7f67717d055a8fd3df2c0/contracts/rate-providers/ASAVAXRateProvider.sol

```
@@ 1,17 @@
18  import { ISAVAX } from '../interfaces/savax/ISAVAX.sol';
19
20  /**
21   * @title Wrapped avax Rate Provider
22   * @notice Returns the hardcoded value of 1e18
23   */
24  contract ASAVAXRateProvider is IRateProvider {
25    ISAVAX public immutable sAVAX;
26
27    constructor(ISAVAX _sAVAX) {
28      sAVAX = _sAVAX;
29    }
30
31    /**
32     * @return the value of sAVAX in terms of AVAX
33     */
34    function getRate() external view override returns (uint256) {
35      return sAVAX.getPooledAvaxByShares(1e18);
36    }
37  }
```

Status

✓ Fixed

[WP-L5] Unnecessary precision loss and gas waste in `TAvaxExchangeRateProvider.getPooledAvaxByShares()`

Low

Issue Description

Current implementation:

$$\left\lfloor \frac{tAVAX.convertToAssets(10^{18}) \times sAVAX.getPooledAvaxByShares(10^{18})}{10^{18}} \right\rfloor$$
$$= \left\lfloor \frac{tAVAX.convertToAssets(10^{18}) \times \left\lfloor \frac{10^{18} \times sAVAX.totalPooledAvax}{sAVAX.totalShares} \right\rfloor}{10^{18}} \right\rfloor$$

Simplified version:

$$\left\lfloor \frac{tAVAX.convertToAssets(10^{18}) \times sAVAX.totalPooledAvax}{sAVAX.totalShares} \right\rfloor$$

```
49    /**
50     * @return the value of tAVAX in terms of AVAX
51     */
52    function getRate() external view returns (uint256) {
53      return (tAVAX.convertToAssets(1e18) * sAVAX.getPooledAvaxByShares(1e18)) /
      1e18;
54    }
```

```
211     /**
212      * @return The amount of AVAX that corresponds to `shareAmount` token shares.
213      */
214     function getPooledAvaxByShares(uint shareAmount) public view returns (uint) {
```

```
215         if (totalShares == 0) {
216             return 0;
217         }
218 
219         return shareAmount.mul(totalPooledAvax).div(totalShares);
220     }
```

Recommendation

Consider changing to something like:

```
49  /**
50   * @return the value of tAVAX in terms of AVAX
51   */
52  function getRate() external view returns (uint256) {
53    return sAVAX.getPooledAvaxByShares(tAVAX.convertToAssets(1e18));
54  }
```

Status

✓ Fixed

[WP-M6] ChainlinkRateProvider fails to account for token decimals and updatedAt (whether the price data is stale)

Medium

Issue Description

Based on the usage of `IRateProvider.getRate()`, the expected return value should be "how many wei AVAX per 1 wei asset" multiplied by 10^{18}, i.e., $\frac{10^{18} \times avaxAmountInWei}{assetAmountInWei}$

The Chainlink price feed's `answer` represents "how many whole B tokens (i.e., $10^{bTokenDecimals}$ wei B tokens) one whole A token (i.e., $10^{aTokenDecimals}$ wei A token) is worth" multiplied by $10^{priceFeed.decimals}$, i.e.,

$$answer = \frac{10^{priceFeed.decimals} \times \frac{bTokenAmountInWei}{10^{bTokenDecimals}}}{\frac{aTokenAmountInWei}{10^{aTokenDecimals}}} = \frac{10^{priceFeed.decimals} \times bTokenAmountInWei \times 10^{aTokenDecima}}{aTokenAmountInWei \times 10^{bTokenDecimals}}$$

After transformation:

$$\frac{answer \times 10^{18} \times 10^{bTokenDecimals}}{10^{priceFeed.decimals} \times 10^{aTokenDecimals}} = \frac{10^{18} \times bTokenAmountInWei}{aTokenAmountInWei}$$

If B token is AVAX and A token is the asset token:

$$\frac{answer \times 10^{18} \times 10^{avaxDecimals}}{10^{priceFeed.decimals} \times 10^{assetDecimals}} = \frac{10^{18} \times avaxAmountInWei}{assetAmountInWei}$$

Conclusion

`ChainlinkRateProvider.getRate()`

- Expected result:

$$\frac{answer \times 10^{18} \times 10^{avaxDecimals}}{10^{priceFeed.decimals} \times 10^{assetDecimals}} = \frac{answer \times 10^{18} \times 10^{18}}{10^{priceFeed.decimals} \times 10^{assetDecimals}}$$

- Current implementation:

$$answer \times \_scalingFactor = answer \times \frac{10^{18}}{10^{priceFeed.decimals}}$$

  - Compared to the expected result, it's missing the $\frac{10^{18}}{10^{assetDecimals}}$ component

The current implementation will only work when `assetDecimals` happens to be 18.

Furthermore, `ChainlinkRateProvider.getRate()` directly uses the `answer` from `pricefeed.latestRoundData()` without validating the `updatedAt` return value, which could lead to using stale or inaccurate price data.

```
@@ 1,13 @@
14
15  pragma solidity ^0.8.0;
16
17  import '@chainlink/contracts/src/v0.8/shared/interfaces/AggregatorV3Interface.sol';
18  import '@chainlink/contracts/src/v0.8/interfaces/FeedRegistryInterface.sol';
19  import './IRateProvider.sol';
20
21  /**
22   * @title Chainlink Rate Provider
23   * @notice Returns a Chainlink price feed's quote for the provided currency pair
24   * @dev This rate provider is a simplification of ChainlinkReistryRateProvider
    which is fixed to a particular pricefeed.
25   *      This is expected to be used in environments where the Chainlink registry
    is not available.
26   */
27  contract ChainlinkRateProvider is IRateProvider {
28    AggregatorV3Interface public immutable pricefeed;
29
30    // Rate providers are expected to respond with a fixed-point value with 18
    decimals
31    // We then need to scale the price feed's output to match this.
32    uint256 internal immutable _scalingFactor;
33
34    /**
35     * @param feed - The Chainlink price feed contract
36     */
37    constructor(AggregatorV3Interface feed) {
38      pricefeed = feed;
39      _scalingFactor = 10 ** (18 - feed.decimals());
```

```
40    }
41
42    /**
43     * @return the value of the quote currency in terms of the base currency
44     */
45    function getRate() external view override returns (uint256) {
46      (, int256 price, , , ) = pricefeed.latestRoundData();
47      require(price > 0, 'Invalid price rate response');
48      return uint256(price) * _scalingFactor;
49    }
50  }
```

```
33    /**
34     * @notice Returns the rate of an asset in avax terms
35     * @param _asset token address, must be the base currency (not quote)
36     * @return _rateInAvax the exchange rate in 1e18
37     */
38    function getRateInAvax(address _asset) external view returns (uint _rateInAvax)
      {
39      if (_asset == WAVAX) return 1e18;
40
41      if (rateProviders[_asset] == address(0)) revert RateProviderNotFound();
42      _rateInAvax = IRateProvider(rateProviders[_asset]).getRate();
43    }
```

```
25    /**
26     * @notice Calculate total NAV of native AVAX + ERC20 tokens for a target
      address
27     * @param _target Address to calculate NAV for
28     * @param _tokens Array of ERC20 token addresses
29     * @return _nav Total NAV in sAVAX terms
30     * @dev Returns raw sAVAX balance if no other assets, otherwise converts AVAX to
      sAVAX shares + raw sAVAX
31     * @dev Silently ignores tokens with zero rates from the registry
32     */
33    function nav(address _target, address[] memory _tokens) external view returns
      (uint _nav) {
34      _nav += _target.balance;
35
```

```
36      uint wip;
37      uint sAVAXBalance;
38      for (uint i; i < _tokens.length; ++i) {
39        wip = IERC20(_tokens[i]).balanceOf(_target);
40
41        if (wip > 0) {
42          if (_tokens[i] == address(sAVAX)) {
43            sAVAXBalance = wip;
44          } else {
45            uint rate = RATE_PROVIDER_REGISTRY.getRateInAvax(_tokens[i]);
46            if (rate > 0) {
47              _nav += Math.mulDiv(rate, wip, 1e18);
48            }
49            // Note: Zero-rate tokens are silently ignored
50          }
51        }
52      }
53      if (_nav == 0) {
54        _nav = sAVAXBalance;
55      } else {
56        _nav = sAVAX.getSharesByPooledAvax(_nav) + sAVAXBalance;
57      }
58    }
```

Status

ⓘ Acknowledged

Appendix

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by WatchPug; however, WatchPug does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Smart Contract technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.