Treehouse AAVE + Spark Audit Report

Feb 25, 2025

Table of Contents

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/tETH-protocol**
Commit	**484f119c4034aee4489ff54d8e916041c3de3ecf**
Language	**Solidity**

Audit Summary

Delivery Date	**Feb 25, 2025**
Audit Methodology	**Static Analysis, Manual Review**
Total Isssues	**3**

[WP-M1] Using `totalCollateralBase` for calculations may not accurately reflect changes in total assets.

Medium

Issue Description

If `RATE_PROVIDER_REGISTRY.getEthInUsd` is not in sync with `aave IPriceOracleGetter(params.oracle).getAssetPrice(vars.currentReserveAddress)`, the calculated wstETH amount will be inaccurate.

During market volatility, USD prices may differ between oracles, while the underlying ETH amount in totalCollateral might not have actually changed.

GenericLogic.sol

```
 vars.assetPrice =
IPriceOracleGetter(params.oracle).getAssetPrice(vars.currentReserveAddress);
```

NavAaveV3.sol

```
28  function nav(address _target, address _lendingPool) external view returns (uint
    _nav) {
29      (uint totalCollateralBase, uint totalDebtBase, , , , ) =
    IPoolV3(_lendingPool).getUserAccountData(_target);
30
31      // 1e8 base
32      uint navInBase = (totalCollateralBase - totalDebtBase);
33
34      unchecked {
35        // nav in eth
36        _nav = (navInBase * 1e10 * PRECISION) /
    RATE_PROVIDER_REGISTRY.getEthInUsd();
37
38        _nav = IwstETH(wstETH).getWstETHByStETH(_nav);
39      }
40    }
```

```
51  function doAccounting(
52      INavRegistry.ModuleParams[][] calldata dynamicModuleParams
53    ) external whenNotPaused onlyOwnerOrExecutor {
54      unchecked {
55        if (block.timestamp < nextWindow) revert StillInWaitingPeriod();
56        nextWindow = (uint64(block.timestamp) + cooldown);
57
58        uint _lastNav = NAV_LENS.lastRecordedProtocolNav();
59        uint _currentNav = NAV_LENS.currentProtocolNav(dynamicModuleParams);
60
61        bool _isPnlPositive = _currentNav > _lastNav;
62        uint _netPnl = _isPnlPositive ? _currentNav - _lastNav : _lastNav -
    _currentNav;
63
64        if (_netPnl > maxPnl()) revert DeviationExceeded();
65
66        if (_isPnlPositive) {
67          uint _fee = (_netPnl * TREEHOUSE_ACCOUNTING.fee()) / PRECISION;
68          _netPnl -= _fee;
69          TREEHOUSE_ACCOUNTING.mark(ITreehouseAccounting.MarkType.MINT, _netPnl,
    _fee);
70        } else {
71          TREEHOUSE_ACCOUNTING.mark(ITreehouseAccounting.MarkType.BURN, _netPnl, 0);
72        }
73      }
74    }
```

```
83  function currentProtocolNav(
84      INavRegistry.ModuleParams[][] calldata dynamicModuleParams
85    ) external view returns (uint _nav) {
86      _nav += vaultNav();
87      uint _stratLen = STRATEGY_STORAGE.getStrategyCount();
88
89      for (uint i; i < _stratLen; ++i) {
90        _nav += strategyNav(i, dynamicModuleParams[i]);
91      }
92    }
```

Recommendation

Query the collateral amount directly and convert it into ETH value respectively, without using USD oracle prices in the conversion.

Status

ⓘ Acknowledged

[WP-M2] In Spark's eMode, `NavAaveV3.nav()` may incorrectly record profits/losses due to eMode oracle functionality.

Medium

Issue Description

When the eMode oracle is enabled in Spark, multiple assets use the same eMode oracle for pricing (see GenericLogic.solL77-81 and GenericLogic.solL116-119), so it cannot reflect the true market prices accurately.

Since `NavAaveV3.nav()` uses results from Spark's `IPoolV3.getUserAccountData()`, the net asset calculation will be distorted.

Note: Aave v3.2 removed the eMode oracle, but Spark's on-chain code still contains eMode oracle functionality.

NavAaveV3.sol

```
@@ 22,27 @@
28    function nav(address _target, address _lendingPool) external view returns (uint
_nav) {
29        (uint totalCollateralBase, uint totalDebtBase, , , , ) =
IPoolV3(_lendingPool).getUserAccountData(_target);
30
31        // 1e8 base
32        uint navInBase = (totalCollateralBase - totalDebtBase);
33
34        unchecked {
35          // nav in eth
36          _nav = (navInBase * 1e10 * PRECISION) /
RATE_PROVIDER_REGISTRY.getEthInUsd();
37
38          _nav = IwstETH(wstETH).getWstETHByStETH(_nav);
39        }
40    }
```

Pool.sol

```
457   function getUserAccountData(
458     address user
459   )
      @@ 460,463 @@
464     returns (
465       uint256 totalCollateralBase,
466       uint256 totalDebtBase,
      @@ 467,470 @@
471     )
472   {
473     return
474       PoolLogic.executeGetUserAccountData(
475         _reserves,
476         _reservesList,
477         _eModeCategories,
478         DataTypes.CalculateUserAccountDataParams({
479           userConfig: _usersConfig[user],
480           reservesCount: _reservesCount,
481           user: user,
482           oracle: ADDRESSES_PROVIDER.getPriceOracle(),
483           userEModeCategory: _usersEModeCategory[user]
484         })
485       );
486   }
```

PoolLogic.sol

```
      @@ 143,155 @@
156   function executeGetUserAccountData(
      @@ 157,160 @@
161   )
162     external
163     view
164     returns (
165       uint256 totalCollateralBase,
166       uint256 totalDebtBase,
```

```
@@ 167,170 @@
    )
  {
    (
      totalCollateralBase,
      totalDebtBase,
      ltv,
      currentLiquidationThreshold,
      healthFactor,

    ) = GenericLogic.calculateUserAccountData(reservesData, reservesList,
eModeCategories, params);

@@ 182,186 @@
  }
```

```
@@ 49,63 @@
  function calculateUserAccountData(
@@ 65,68 @@
  ) internal view returns (uint256, uint256, uint256, uint256, uint256, bool) {
    if (params.userConfig.isEmpty()) {
      return (0, 0, 0, 0, type(uint256).max, false);
    }

    CalculateUserAccountDataVars memory vars;

    if (params.userEModeCategory != 0) {
      (vars.eModeLtv, vars.eModeLiqThreshold, vars.eModeAssetPrice) = EModeLogic
        .getEModeConfiguration(
          eModeCategories[params.userEModeCategory],
          IPriceOracleGetter(params.oracle)
        );
    }

    while (vars.i < params.reservesCount) {
      if (!params.userConfig.isUsingAsCollateralOrBorrowing(vars.i)) {
        unchecked {
          ++vars.i;
        }
```

```
89          continue;
90        }
91
92        vars.currentReserveAddress = reservesList[vars.i];
93
94        if (vars.currentReserveAddress == address(0)) {
95          unchecked {
96            ++vars.i;
97          }
98          continue;
99        }
100
101       DataTypes.ReserveData storage currentReserve =
reservesData[vars.currentReserveAddress];
102
103       (
@@ 104,109 @@
110       ) = currentReserve.configuration.getParams();
111
112       unchecked {
113         vars.assetUnit = 10 ** vars.decimals;
114       }
115
116       vars.assetPrice = vars.eModeAssetPrice != 0 &&
117         params.userEModeCategory == vars.eModeAssetCategory
118         ? vars.eModeAssetPrice
119         :
IPriceOracleGetter(params.oracle).getAssetPrice(vars.currentReserveAddress);
120
121       if (vars.liquidationThreshold != 0 &&
params.userConfig.isUsingAsCollateral(vars.i)) {
122         vars.userBalanceInBaseCurrency = _getUserBalanceInBaseCurrency(
123           params.user,
124           currentReserve,
125           vars.assetPrice,
126           vars.assetUnit
127         );
128
129         vars.totalCollateralInBaseCurrency += vars.userBalanceInBaseCurrency;
130
131         vars.isInEModeCategory = EModeLogic.isInEModeCategory(
132           params.userEModeCategory,
```

```
133           vars.eModeAssetCategory
134         );
135
136         if (vars.ltv != 0) {
@@ 137,141 @@
142         }
143
144         vars.avgLiquidationThreshold +=
145           vars.userBalanceInBaseCurrency *
146           (vars.isInEModeCategory ? vars.eModeLiqThreshold :
vars.liquidationThreshold);
147       }
148
149       if (params.userConfig.isBorrowing(vars.i)) {
150         vars.totalDebtInBaseCurrency += _getUserDebtInBaseCurrency(
@@ 151,154 @@
155         );
156       }
157
158       unchecked {
159         ++vars.i;
160       }
161     }
162
163     unchecked {
164       vars.avgLtv = vars.totalCollateralInBaseCurrency != 0
165         ? vars.avgLtv / vars.totalCollateralInBaseCurrency
166         : 0;
167       vars.avgLiquidationThreshold = vars.totalCollateralInBaseCurrency != 0
168         ? vars.avgLiquidationThreshold / vars.totalCollateralInBaseCurrency
169         : 0;
170     }
171
172     vars.healthFactor = (vars.totalDebtInBaseCurrency == 0)
173       ? type(uint256).max
174       :
(vars.totalCollateralInBaseCurrency.percentMul(vars.avgLiquidationThreshold)).wadDiv(
175         vars.totalDebtInBaseCurrency
176       );
177     return (
```

```
@@ 178,183 @@
184       );
185     }
```

Recommendation

See Recommendation of [WP-M1].

Status

ⓘ Acknowledged

[WP-L3] `AaveV3Withdraw#_withdraw()` using `type(uint).max` for the whole amount will return the wrong `withdrawnAmount` .

Low

Issue Description

Per the comments, `AaveV3Withdraw` is designed to support passing `type(uint).max` as the amount for the whole amount; however, the returned amount should be the amount that was actually withdrawn.

AaveV3Withdraw.sol

```
50    /// @notice User withdraws tokens from the Aave protocol
51    /// @param _assetId The id of the token to be deposited
52    /// @param _amount Amount of tokens to be withdrawn -> send type(uint).max for
      whole amount
53    /// @param _poolId The id of the pool
54    function _withdraw(uint16 _assetId, uint _amount, uint16 _poolId) internal
      returns (uint, bytes memory) {
55        address _lendingPool =
      IProtocolPoolController(PROTOCOL_CONTROLLER).getPoolAddress(PROTOCOL_ID, _poolId);
56        address tokenAddr = IPoolV3(_lendingPool).getReserveAddressById(_assetId);
57        IPoolV3(_lendingPool).withdraw(tokenAddr, _amount, address(this));
58        bytes memory logData = abi.encode(tokenAddr, _amount);
59        return (_amount, logData);
60    }
```

```
94    /**
95       * @notice Implements the withdraw feature. Through `withdraw()`, users redeem
      their aTokens for the underlying asset
96       * previously supplied in the Aave protocol.
97       * @dev Emits the `Withdraw()` event.
98       * @dev If the user withdraws everything, `ReserveUsedAsCollateralDisabled()` is
      emitted.
99       * @param reservesData The state of all the reserves
100      * @param reservesList The addresses of all the active reserves
101      * @param eModeCategories The configuration of all the efficiency mode
      categories
```

```
102     * @param userConfig The user configuration mapping that tracks the
    supplied/borrowed assets
103     * @param params The additional parameters needed to execute the withdraw
    function
104     * @return The actual amount withdrawn
105     */
106   function executeWithdraw(
107     mapping(address => DataTypes.ReserveData) storage reservesData,
108     mapping(uint256 => address) storage reservesList,
109     mapping(uint8 => DataTypes.EModeCategory) storage eModeCategories,
110     DataTypes.UserConfigurationMap storage userConfig,
111     DataTypes.ExecuteWithdrawParams memory params
112   ) external returns (uint256) {
113     DataTypes.ReserveData storage reserve = reservesData[params.asset];
114     DataTypes.ReserveCache memory reserveCache = reserve.cache();
115
116     require(params.to != reserveCache.aTokenAddress, Errors.WITHDRAW_TO_ATOKEN);
117
118     reserve.updateState(reserveCache);
119
120     uint256 userBalance =
    IAToken(reserveCache.aTokenAddress).scaledBalanceOf(msg.sender).rayMul(
121       reserveCache.nextLiquidityIndex
122     );
123
124     uint256 amountToWithdraw = params.amount;
125
126     if (params.amount == type(uint256).max) {
127       amountToWithdraw = userBalance;
128     }
129
130     ValidationLogic.validateWithdraw(reserveCache, amountToWithdraw, userBalance);
131
132     reserve.updateInterestRatesAndVirtualBalance(reserveCache, params.asset, 0,
    amountToWithdraw);
133
134     bool isCollateral = userConfig.isUsingAsCollateral(reserve.id);
135
136     if (isCollateral && amountToWithdraw == userBalance) {
137       userConfig.setUsingAsCollateral(reserve.id, false);
138       emit ReserveUsedAsCollateralDisabled(params.asset, msg.sender);
139     }
140
```

```
141       IAToken(reserveCache.aTokenAddress).burn(
142         msg.sender,
143         params.to,
144         amountToWithdraw,
145         reserveCache.nextLiquidityIndex
146       );
147
148       if (isCollateral && userConfig.isBorrowingAny()) {
149         ValidationLogic.validateHFAndLtv(
150           reservesData,
151           reservesList,
152           eModeCategories,
153           userConfig,
154           params.asset,
155           msg.sender,
156           params.reservesCount,
157           params.oracle,
158           params.userEModeCategory
159         );
160       }
```

Status

✓ Fixed

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.