Treehouse Do Accounting Generalization 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	**6**

[WP-M1] PnlAccounting uses incorrect precision when applying TREEHOUSE_ACCOUNTING.fee

Medium

Issue Description

For example:

When:

- TREEHOUSE_ACCOUNTING.fee: 100

Expected:

- Fee rate should be 100 / TREEHOUSE_ACCOUNTING.PRECISION = 100 / 1e4 = 1%

Current implementation:

- Fee rate is 100 / PnlAccounting.PRECISION = 100 / 1e6 = 0.01%

```
16  contract PnlAccounting is Ownable2Step, Pausable {
17    uint constant PRECISION = 1e6;
18
    @@ 19,46 @@
47
48    /**
49     * @notice mark to market protocol NAV
50     */
51    function doAccounting(
52      INavRegistry.ModuleParams[][] calldata dynamicModuleParams
53    ) external whenNotPaused onlyOwnerOrExecutor {
54      unchecked {
    @@ 55,64 @@
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      }
75
@@ 76,135 @@
136   }
```

```
33    contract TreehouseAccounting is ITreehouseAccounting, Ownable2Step {
34      using SafeERC20 for IERC20;
35      uint16 constant PRECISION = 1e4;
36
@@ 37,40 @@
41      uint16 public fee; // in bips
42
43      constructor(
@@ 44,48 @@
49        uint16 _fee
50      ) Ownable(_creator) {
@@ 51,54 @@
55        fee = _fee;
56
57        IERC20(IAU).approve(address(TASSET), type(uint).max);
58      }
59
@@ 60,101 @@
102
103     /**
104      * @notice Set the protocol fee
105      * @param _newFee new fee in bips
106      */
107     function setFee(uint16 _newFee) external onlyOwner {
108       if (_newFee > PRECISION) revert('max_fee');
109       emit FeeUpdated(_newFee, fee);
110       fee = _newFee;
111     }
112   }
```

Status

✓ **Fixed**

[WP-M2] Incorrect calculation of `nextWindow` in `doAccounting` function

Medium

Issue Description

PnlAccounting.solL56 incorrectly uses `+=` instead of `=`, causing `nextWindow` to be unreachable.

```
48    /**
49     * @notice mark to market protocol NAV
50     */
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    }
```

Recommendation

```
48    /**
49     * @notice mark to market protocol NAV
50     */
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    }
```

Status

✓ Fixed

[WP-L3] `setDeviation` should check `_newDeviation`

Low

Issue Description

The current implementation incorrectly validates the previous `deviation` value instead of the new input.

```
131   function setDeviation(uint16 _newDeviation) external onlyOwner {
132       if (deviation > PRECISION) revert DeviationExceeded();
133       emit DeviationUpdated(_newDeviation, deviation);
134       deviation = _newDeviation;
135     }
```

Status

✓ Fixed

[WP-I4] Only the old module address is stored, making `revertModule` unable to recover the module name.

Informational

Issue Description

`revertModule(id)` only reverts the address but not the name; it may result in a misleading name for the module.

```
60  mapping(bytes4 moduleId => address) public previousModuleAddresses;
```

```
120   /**
121    * @notice Starts an address change for an existing entry
122    * @param id Id of contract
123    * @param newAddr address of the new contract
124    * @param name human readable module name
125    */
126   function updateModule(bytes4 id, address newAddr, string calldata name) external
      onlyOwner {
127     if (!modules[id].exists) {
128       revert EntryNonExistentError(id);
129     }
130     address _oldAddr = modules[id].addr;
131     previousModuleAddresses[id] = _oldAddr;
132     modules[id].addr = newAddr;
133     modules[id].name = name;
134
135     emit UpdateContract(id, newAddr, _oldAddr);
136   }
137
138   /**
139    * @notice reverts to the previous address immediately
140    * @dev In case the new version has a fault, a quick way to fallback to the old
      contract
141    * @param id Id of contract
142    */
143   function revertModule(bytes4 id) external onlyOwner {
144     if (!(modules[id].exists)) {
```

```
145         revert EntryNonExistentError(id);
146       }
147       if (previousModuleAddresses[id] == address(0)) {
148         revert EmptyPrevAddrError(id);
149       }
150
151       address currentAddr = modules[id].addr;
152       modules[id].addr = previousModuleAddresses[id];
153
154       emit RevertToPreviousAddress(id, modules[id].addr, currentAddr);
155     }
```

Status

ⓘ Acknowledged

[WP-I5] `NavRegistry.getStrategyNav()` should revert when the `info` length returned by staticcall is less than 32 bytes.

Informational

Issue Description

Currently, when an invalid or non-contract module address is provided, the implementation still treats it as a successful call because a low-level `staticcall` is used.

Reverting in such cases would help detect errors earlier in the execution flow.

```
214     /**
215      * @notice Get Nav of a strategy
216      *
217      * @dev loop `moduleIds.length` times, if module is dynamic,
218      * loop dynamicModuleParams to retrieve first instance of moduleId + cd.
219      * dynamicModuleParams must not have duplicates.
220      *
221      * @param strategy strategy address
222      * @param dynamicModuleParams array of dynamic module params
223      * @return _navInUnderlying sum of all attached modules
224      */
225     function getStrategyNav(
226       address strategy,
227       ModuleParams[] calldata dynamicModuleParams
228     ) external view returns (uint _navInUnderlying) {
229       bytes32[] memory moduleIds = _strategyModuleIds[strategy].moduleIds.values();
230
231       for (uint i; i < moduleIds.length; ++i) {
232         if (bytes32(strategyModuleCd[strategy][bytes4(moduleIds[i])]) != DYNAMIC) {
233           (bool success, bytes memory info) =
    address(modules[bytes4(moduleIds[i])].addr).staticcall(
234             strategyModuleCd[strategy][bytes4(moduleIds[i])]
235           );
236
237           if (!success) revert GetNavFailed(bytes4(moduleIds[i]));
238
239           unchecked {
240             _navInUnderlying += uint(bytes32(info));
241           }
```

```
242         } else {
243           // get first instance of dynamic cd
244           bool executed = false;
245
246           for (uint j; j < dynamicModuleParams.length; ++j) {
247             if (dynamicModuleParams[j].moduleId == moduleIds[i]) {
248               (bool success, bytes memory info) =
    address(modules[bytes4(moduleIds[i])].addr).staticcall(
249                 dynamicModuleParams[j].cd
250               );
251               if (!success) revert GetNavFailed(bytes4(moduleIds[i]));
252
253               unchecked {
254                 _navInUnderlying += uint(bytes32(info));
255               }
256
257               executed = true;
258               break;
259             }
260           }
261
262           if (!executed) revert MissingDynamicModule(bytes4(moduleIds[i]));
263         }
264       }
265     }
```

Status

ⓘ Acknowledged

[WP-G6] Using `values(struct EnumerableSet.Bytes32Set set)` in state-modifying business logic (non-view functions) unnecessarily wastes gas

Gas

Issue Description

```
211     /**
212      * @dev Return the entire set in an array
213      *
214      * WARNING: This operation will copy the entire storage to memory, which can
    be quite expensive. This is designed
215      * to mostly be used by view accessors that are queried without any gas fees.
    Developers should keep in mind that
216      * this function has an unbounded cost, and using it as part of a
    state-changing function may render the function
217      * uncallable if the set grows to a point where copying to memory consumes too
    much gas to fit in a block.
218      */
219     function values(Bytes32Set storage set) internal view returns (bytes32[]
    memory) {
220         bytes32[] memory store = _values(set._inner);
221         bytes32[] memory result;
222
223         /// @solidity memory-safe-assembly
224         assembly {
225             result := store
226         }
227
228         return result;
229     }
```

caller

https://github.com/treehouse-gaia/tETH-protocol/blob/24525b2dfa9d1b0af375b8db171bfdd96c604f79/contracts/periphery/PnlAccounting.sol#L51-L74

```
51    function doAccounting(
52      INavRegistry.ModuleParams[][] calldata dynamicModuleParams
53    ) external whenNotPaused onlyOwnerOrExecutor {
54      unchecked {
@@ 55,56 @@
57
58        uint _lastNav = NAV_LENS.lastRecordedProtocolNav();
59        uint _currentNav = NAV_LENS.currentProtocolNav(dynamicModuleParams);
60
@@ 61,72 @@
73      }
74    }
```

https://github.com/treehouse-gaia/tETH-protocol/blob/24525b2dfa9d1b0af375b8db171bfdd96c604f79/contracts/periphery/NavLens.sol#L77-L86

```
77    function currentProtocolNav(
78      INavRegistry.ModuleParams[][] calldata dynamicModuleParams
79    ) external view returns (uint _nav) {
80      _nav += vaultNav();
81      uint _stratLen = STRATEGY_STORAGE.getStrategyCount();
82
83      for (uint i; i < _stratLen; ++i) {
84        _nav += strategyNav(i, dynamicModuleParams[i]);
85      }
86    }
```

https://github.com/treehouse-gaia/tETH-protocol/blob/24525b2dfa9d1b0af375b8db171bfdd96c604f79/contracts/periphery/NavLens.sol#L59-L64

```
59    function strategyNav(
60      uint strategyId,
61      INavRegistry.ModuleParams[] calldata dynamicModuleParams
62    ) public view returns (uint) {
63      return
NAV_REGISTRY.getStrategyNav(STRATEGY_STORAGE.getStrategyAddress(strategyId),
dynamicModuleParams);
64    }
```

https://github.com/treehouse-gaia/tETH-protocol/blob/24525b2dfa9d1b0af375b8db171bfdd96c604f79/contracts/NavRegistry.sol#L225-L265

```
225   function getStrategyNav(
226     address strategy,
227     ModuleParams[] calldata dynamicModuleParams
228   ) external view returns (uint _navInUnderlying) {
229     bytes32[] memory moduleIds = _strategyModuleIds[strategy].moduleIds.values();
230
231     for (uint i; i < moduleIds.length; ++i) {
232       if (bytes32(strategyModuleCd[strategy][bytes4(moduleIds[i])]) != DYNAMIC) {
233         (bool success, bytes memory info) =
      address(modules[bytes4(moduleIds[i])].addr).staticcall(
234           strategyModuleCd[strategy][bytes4(moduleIds[i])]
235         );
236
237         if (!success) revert GetNavFailed(bytes4(moduleIds[i]));
238
239         unchecked {
240           _navInUnderlying += uint(bytes32(info));
241         }
242       } else {
243         // get first instance of dynamic cd
244         bool executed = false;
245
246         for (uint j; j < dynamicModuleParams.length; ++j) {
247           if (dynamicModuleParams[j].moduleId == moduleIds[i]) {
248             (bool success, bytes memory info) =
      address(modules[bytes4(moduleIds[i])].addr).staticcall(
249               dynamicModuleParams[j].cd
250             );
251             if (!success) revert GetNavFailed(bytes4(moduleIds[i]));
252
253             unchecked {
254               _navInUnderlying += uint(bytes32(info));
255             }
256
257             executed = true;
258             break;
259           }
260         }
261
262         if (!executed) revert MissingDynamicModule(bytes4(moduleIds[i]));
```

```
263         }
264       }
265     }
```

Recommendation

Consider using `length(struct EnumerableSet.Bytes32Set set)` and `at(struct EnumerableSet.Bytes32Set set, uint256 index)` :

```
225     function getStrategyNav(
226       address strategy,
227       ModuleParams[] calldata dynamicModuleParams
228     ) external view returns (uint _navInUnderlying) {
229       uint256 moduleIdsLength = _strategyModuleIds[strategy].moduleIds.length();
230       for (uint i; i < moduleIdsLength; ++i) {
231         bytes4 moduleId = bytes4(_strategyModuleIds[strategy].moduleIds.at(i));
232         if (bytes32(strategyModuleCd[strategy][moduleId]) != DYNAMIC) {
233           (bool success, bytes memory info) =
address(modules[moduleId].addr).staticcall(
234             strategyModuleCd[strategy][moduleId]
235           );
236
237           if (!success) revert GetNavFailed(moduleId);
238
239           unchecked {
240             _navInUnderlying += uint(bytes32(info));
241           }
242         } else {
243           // get first instance of dynamic cd
244           bool executed = false;
245
246           for (uint j; j < dynamicModuleParams.length; ++j) {
247             if (dynamicModuleParams[j].moduleId == moduleId) {
248               (bool success, bytes memory info) =
address(modules[moduleId].addr).staticcall(
249                 dynamicModuleParams[j].cd
250               );
251               if (!success) revert GetNavFailed(moduleId);
252
253               unchecked {
254                 _navInUnderlying += uint(bytes32(info));
```

```
255                 }
256
257                 executed = true;
258                 break;
259               }
260             }
261
262             if (!executed) revert MissingDynamicModule(moduleId);
263           }
264         }
265       }
```

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.