Treehouse tETH Timelock Audit Report

May 23, 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	**427d98bdbfee7b47e24ee75681f6054152ac120f**
Language	**Solidity**

Audit Summary

Delivery Date	**May 23, 2025**
Audit Methodology	**Static Analysis, Manual Review**
Total Isssues	**4**

[WP-M1] Consider using SafeERC20

Medium

Issue Description

Non-standard IERC20 tokens (like `USDT` which has no return value) will revert when using `IERC20.transfer()` directly, causing these tokens to be stuck in the `VaultRescuer` contract.

```
58    /**
59     * Allows owner to withdraw funds to `FUNDS_RECEIVER` after `WAIT_TIME` has
    passed, since the rescue
60     */
61    function withdrawFunds(IERC20 tokenContract) external onlyOwner {
62      if (block.timestamp < lastRescuedTimestamp + WAIT_TIME) revert
    TimelockInEffect();
63
64      uint balance;
65      bool success;
66
67      if (address(tokenContract) == ETH) {
68        balance = address(this).balance;
69        (success, ) = address(FUNDS_RECEIVER).call{ value: balance }('');
70      } else {
71        balance = IERC20(tokenContract).balanceOf(address(this));
72        success = tokenContract.transfer(FUNDS_RECEIVER, balance);
73      }
74
75      if (!success) revert WithdrawFailed();
76      emit FundsWithdrawn(address(tokenContract), balance);
77    }
```

Recommendation

Consider using OpenZeppelin SafeERC20, similar to `contracts/libs/Rescuable.sol#rescueERC20()` .

Status

✓ Fixed

[WP-L2] To avoid confusion on chains where the native token is not ETH (e.g., Polygon), consider using ”Native” instead of ”ETH” when referring to native tokens.

Low

Issue Description

For example, consider:

- `IRescuable.rescueETH()` -> `IRescuable.rescueNative()`
- `VaultRescuer.rescueETH()` -> `VaultRescuer.rescueNative()`
- `VaultRescuer.ETH = 0xEeeeeEeeeEeEeeEeEeEeeEEEeeeeEeeeeeeeEEeE;` -> `VaultRescuer.NATIVE = address(0)`

```
7   interface IRescuable {
8     function rescueETH(address to) external;
9
10    function rescueERC20(IERC20 tokenContract, address to, uint amount) external;
11  }
12
13  /**
14   * This contract allows rescuing from the vault into a hardcoded funds receiver
    address
15   */
16  contract VaultRescuer is Ownable2Step {
17    error TimelockInEffect();
18    error WithdrawFailed();
19
20    event FundsRescued(address token, uint amount);
21    event FundsWithdrawn(address token, uint amount);
22
23    address public immutable FUNDS_RECEIVER;
24    address public immutable VAULT;
25    uint public immutable WAIT_TIME = 5 days;
26    address public constant ETH = 0xEeeeeEeeeEeEeeEeEeEeeEEEeeeeEeeeeeeeEEeE;
27
28    uint public lastRescuedTimestamp;
29
```

```
@@ 30,45 @@

  /** Rescue native ETH from vault */
  function rescueETH() external onlyOwner {
    uint balance = address(this).balance;
    IRescuable(VAULT).rescueETH(address(this));
    balance = address(this).balance - balance;

    lastRescuedTimestamp = block.timestamp;

    emit FundsRescued(ETH, balance);
  }

  /**
   * Allows owner to withdraw funds to `FUNDS_RECEIVER` after `WAIT_TIME` has passed, since the rescue
   */
  function withdrawFunds(IERC20 tokenContract) external onlyOwner {
    if (block.timestamp < lastRescuedTimestamp + WAIT_TIME) revert TimelockInEffect();

    uint balance;
    bool success;

    if (address(tokenContract) == ETH) {
      balance = address(this).balance;
      (success, ) = address(FUNDS_RECEIVER).call{ value: balance }('');
    } else {
      balance = IERC20(tokenContract).balanceOf(address(this));
      success = tokenContract.transfer(FUNDS_RECEIVER, balance);
    }

    if (!success) revert WithdrawFailed();
    emit FundsWithdrawn(address(tokenContract), balance);
  }

  receive() external payable {}
}
```

Status

✓ Fixed

[WP-I3] `VAULT.rescueERC20()` and `VAULT.rescueETH()` Should Not Impact `currentProtocolNav()`

Informational

Issue Description

Assets in the Vault are part of `NavLens.currentProtocolNav()`. If `VaultRescuer` and `FUNDS_RECEIVER` are not counted as part of `currentProtocolNav()`, when `VaultRescuer.rescueERC20()` or `VaultRescuer.rescueETH()` is executed, `currentProtocolNav()` will suddenly drop, affecting the PnL in the next settlement.

The rescued assets should not be considered a loss.

`currentProtocolNav()` may need to handle assets in `VaultRescuer` and `FUNDS_RECEIVER`.

```
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    }
```

```
79    /**
80     * current protocol NAV
81     * @param dynamicModuleParams dynamic NAV module metadata
82     */
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    }
```

```
51    /**
52     * @notice vault NAV in terms of the underlying asset
53     */
54    function vaultNav() public view returns (uint) {
55      address erc20NavModule = NAV_REGISTRY.getModuleAddress(0x7bc1fd06);
56      if (erc20NavModule == address(0)) revert NavModuleNotSet();
57
58      return INavErc20(erc20NavModule).nav(VAULT,
    IVault(VAULT).getAllowableAssets());
59    }
```

```
31    /**
32     * @notice get sum of `_tokens` + native token NAV of `_target`
33     * @param _target address to target
34     * @param _tokens token array to price
35     * @return _nav NAV in wstETH terms
36     */
```

```
37    function nav(address _target, address[] memory _tokens) external view returns
(uint _nav) {
38      _nav += _target.balance;
39
40      uint wip;
41      uint wstETHBalance;
42      for (uint i; i < _tokens.length; ++i) {
43        wip = IERC20(_tokens[i]).balanceOf(_target);
44
45        if (wip > 0) {
46          unchecked {
47            if (_tokens[i] == address(wstETH)) {
48              wstETHBalance = wip;
49            } else if (_tokens[i] == address(wstETH.stETH())) {
50              _nav += wip;
51            } else {
52              _nav += (RATE_PROVIDER_REGISTRY.getRateInEth(_tokens[i]) * wip) /
1e18;
53            }
54          }
55        }
56      }
57
58      _nav = wstETH.getWstETHByStETH(_nav) + wstETHBalance;
59    }
```

https://github.com/treehouse-gaia/tETH-protocol/blob/f8fd9dddc824c7b2859e86a5e2e49bceff55d585/contracts/periphery/VaultRescuer.sol#L13-L80

```
13  /**
14   * This contract allows rescuing from the vault into a hardcoded funds receiver
address
15   */
16  contract VaultRescuer is Ownable2Step {
    @@ 17,33 @@
34
35    /** Rescue ERC20 from vault */
36    function rescueERC20(IERC20 tokenContract, uint amount) external onlyOwner {
37      if (amount == 0) {
38        amount = IERC20(tokenContract).balanceOf(VAULT);
```

```
39      }
40
41      IRescuable(VAULT).rescueERC20(tokenContract, address(this), amount);
42      lastRescuedTimestamp = block.timestamp;
43
44      emit FundsRescued(address(tokenContract), amount);
45    }
46
47    /** Rescue native ETH from vault */
48    function rescueETH() external onlyOwner {
49      uint balance = address(this).balance;
50      IRescuable(VAULT).rescueETH(address(this));
51      balance = address(this).balance - balance;
52
53      lastRescuedTimestamp = block.timestamp;
54
55      emit FundsRescued(ETH, balance);
56    }
57

@@ 58,79 @@

80  }
```

Status

ⓘ Acknowledged

[WP-I4] When `amount` is 0 and `VAULT` has no tokens, the timelock will still be reset.

Informational

Issue Description

```
35    /** Rescue ERC20 from vault */
36    function rescueERC20(IERC20 tokenContract, uint amount) external onlyOwner {
37      if (amount == 0) {
38        amount = IERC20(tokenContract).balanceOf(VAULT);
39      }
40
41      IRescuable(VAULT).rescueERC20(tokenContract, address(this), amount);
42      lastRescuedTimestamp = block.timestamp;
43
44      emit FundsRescued(address(tokenContract), amount);
45    }
```

Recommendation

```
35    /** Rescue ERC20 from vault */
36    function rescueERC20(IERC20 tokenContract, uint amount) external onlyOwner {
37      if (amount == 0) {
38        amount = IERC20(tokenContract).balanceOf(VAULT);
39      }
40
41      if (amount == 0) {
42          return;
43      }
44
45      IRescuable(VAULT).rescueERC20(tokenContract, address(this), amount);
46      lastRescuedTimestamp = block.timestamp;
47
48      emit FundsRescued(address(tokenContract), amount);
49    }
```

Similarly, `rescueETH` has the same issue.

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.