Treehouse Merkl Claim Audit Report

Apr 23, 2025

Table of Contents

Summary 2

Overview 3

Issues 4

[WP-M1] `_claim()` may revert due to insufficient amount to be transferred. 4

[WP-I2] Wrong comments 6

[WP-G3] Unnecessary calldata to memory copy 7

[WP-N4] Irrelevant/misleading comments 9

Appendix 10

Disclaimer 11

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	**40ae7392d4390d28bf7344831baab6ef6011dab6**
Language	**Solidity**

Audit Summary

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

[WP-M1] `_claim()` may revert due to insufficient amount to be transferred.

Medium

Issue Description

The `_claim()` function expects to receive exactly the amount specified in the parameters of the `MERKL_DISTRIBUTOR.claim()` call. See MerklClaim.solL79.

```
68    function _claim(
69      address[] memory users,
70      address[] memory tokens,
71      uint256[] memory amounts,
72      bytes32[][] memory proofs,
73      bool[] memory toTreasury
74    ) internal returns (uint, bytes memory) {
75      MERKL_DISTRIBUTOR.claim(users, tokens, amounts, proofs);
76
77      for (uint i; i < tokens.length; ++i) {
78        if (toTreasury[i]) {
79          IERC20(tokens[i]).safeTransfer(TREASURY, amounts[i]);
80        }
81      }
82
83      bytes memory logData = abi.encode(tokens, amounts, toTreasury);
84      return (0, logData);
85    }
```

The actual amount of tokens claimed will be less than the amounts in the params when the same token has been claimed before:

```
167   function claim(
168       address[] calldata users,
169       address[] calldata tokens,
170       uint256[] calldata amounts,
171       bytes32[][] calldata proofs
172   ) external {
173       uint256 usersLength = users.length;
```

```
174     if (
175         usersLength == 0 ||
176         usersLength != tokens.length ||
177         usersLength != amounts.length ||
178         usersLength != proofs.length
179     ) revert InvalidLengths();
180 
181     for (uint256 i; i < usersLength; ) {
182         address user = users[i];
183         address token = tokens[i];
184         uint256 amount = amounts[i];
185 
186         // Only approved operator can claim for `user`
187         if (msg.sender != user && tx.origin != user && operators[user][msg.sender]
    == 0) revert NotWhitelisted();
188 
189         // Verifying proof
190         bytes32 leaf = keccak256(abi.encode(user, token, amount));
191         if (!_verifyProof(leaf, proofs[i])) revert InvalidProof();
192 
193         // Closing reentrancy gate here
194         uint256 toSend = amount - claimed[user][token].amount;
195         claimed[user][token] = Claim(SafeCast.toUint208(amount),
    uint48(block.timestamp), getMerkleRoot());
196 
197         IERC20(token).safeTransfer(user, toSend);
198         emit Claimed(user, token, toSend);
199         unchecked {
200             ++i;
201         }
202     }
203 }
```

Status

✓ Fixed

[WP-I2] Wrong comments

Informational

Issue Description

- Not `only recipient of rewards can claim` , an authorized operator can also claim
- `tokens sent to treasury are contingent on the claimable amounts PASSED IN as external params` , only for the first claim of the token, otherwise it will not be the same.

```
8  /// @title Action for claiming Merkl rewards
9  /**
10  * @dev Assumptions:
11  * - only recipient of rewards can claim
12  * - length size of inputs ARE checked in merkl distributor
13  * - tokens and amounts are encoded in the proofs; if the token amounts are wrong code will revert
14  * - tokens sent to treasury are contingent on the claimable amounts PASSED IN as external params
15  */
16 contract MerklClaim is ActionBase {
```

Status

✓ Fixed

[WP-G3] Unnecessary calldata to memory copy

Gas

Issue Description

https://docs.soliditylang.org/en/v0.8.29/types.html#reference-types

> An assignment or type conversion that changes the data location will always incur an automatic copy operation

https://docs.soliditylang.org/en/v0.8.29/types.html#data-location

> If you can, try to use `calldata` as data location because it will avoid copies and also makes sure that the data cannot be modified. Arrays and structs with `calldata` data location can also be returned from functions, but it is not possible to allocate such types.

Consider changing the data location of `parseInputs(bytes memory _callData)` parameter to `calldata` .

https://github.com/treehouse-gaia/tETH-protocol/blob/40ae7392d4390d28bf7344831baab6ef6011dab6/contracts/strategy/actions/merkl/MerklClaim.sol#L43-L101

```
43    function executeAction(
44      bytes calldata _callData,
45      uint8[] memory,
46      bytes32[] memory
47    ) public payable virtual override returns (bytes32) {
48      Params memory params = parseInputs(_callData);
@@ 49,57 @@
58    }
59
@@ 60,85 @@
86
87    function parseInputs(bytes memory _callData) public pure returns (Params memory
params) {
88      (
```

```
89          address[] memory users,
90          address[] memory tokens,
91          uint256[] memory amounts,
92          bytes32[][] memory proofs,
93          bool[] memory toTreasury
94      ) = abi.decode(_callData, (address[], address[], uint256[], bytes32[][],
    bool[]));
95
    @@ 96,100 @@
101     }
```

```
24    function executeAction(
25      bytes calldata _callData,
26      uint8[] memory,
27      bytes32[] memory
28    ) public payable virtual override returns (bytes32) {
29      Params memory params = parseInputs(_callData);
    @@ 30,32 @@
33    }
34
    @@ 35,44 @@
45
46    function parseInputs(bytes memory _callData) public pure returns (Params memory
    params) {
47      params = abi.decode(_callData, (Params));
48    }
```

Status

✓ Fixed

[WP-N4] Irrelevant/misleading comments

Issue Description

https://github.com/treehouse-gaia/tETH-protocol/blob/40ae7392d4390d28bf7344831baab6ef6011dab6/contracts/strategy/actions/merkl/MerklToggleOperator.sol#L37-L44

```
37    /// @notice User deposits tokens into gearbox pool
38    /// @param operator The whitelisted operator that can claim rewards
39    function _toggle(address operator) internal returns (bool isAuthorized, bytes memory) {
@@ 40,43 @@
44    }
```

https://github.com/treehouse-gaia/tETH-protocol/blob/40ae7392d4390d28bf7344831baab6ef6011dab6/contracts/strategy/actions/merkl/MerklClaim.sol#L62-L85

```
62    /// @notice claim gearbox tokens
63    /// @param users Recipient of tokens
64    /// @param tokens ERC20 claimed
65    /// @param amounts Amount of tokens that will be sent to the corresponding users
66    /// @param proofs Array of hashes bridging from a leaf `(hash of user | token | amount)` to the Merkle root
67    /// @param toTreasury Whether to send the claimed tokens to treasury
68    function _claim(
@@ 69,73 @@
74    ) internal returns (uint, bytes memory) {
@@ 75,84 @@
85    }
```

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.