Sympie

Security Audit Report

July 31, 2024

Contents

1 Introduction

1.1 About Sympie

Developed by `Magpie`, `Sympie` is a DeFi platform that enables participants to enhance their rewards and flexibility through liquid restaking. Users can deposit their assets on `Sympie` to validate new services within the `Symbiotic` shared security framework. In return, they receive LRTs (Liquid Restaked Tokens), which are liquid restaked versions of the deposited assets. This allows users to access new yield opportunities in DeFi while their underlying tokens continue to generate rewards, without being locked up. Importantly, LRTs issued by `Sympie` are tradable and transferable, enhancing liquidity and usability for participants.

1.2 Audit Scope

The following source code was reviewed during the audit:

- https://github.com/magpiexyz/sympie_contract/pull/1
- Commit ID: 8aac06a

And this is the final version representing all fixes implemented for the issues identified in the audit:

- https://github.com/magpiexyz/sympie_contract/pull/1
- Commit ID: 977a124

Note this audit only covers the `SympieStaking.sol`, `SympiePreDepositHelper.sol`, `SympieConfig.sol`, `PriceProvider.sol`, `MLRT.sol`, `TransferHelper.sol`, and `SympieConfigRoleChecker.sol` contracts.

1.3 Changelog

Version	Date
First Audit	July 28, 2024

2 | Overall Assessment

This report has been compiled to identify issues and vulnerabilities within the `Sympie` project. Throughout this audit, we identified several issues spanning various severity levels. By employing auxiliary tool techniques to supplement our thorough manual code review, we have discovered the following findings.

Severity	Count	Acknowledged	Won't Do	Addressed
Critical	-	-	-	-
High	1	-	-	1
Medium	1	-	-	1
Low	1	1	-	-
Informational	-	-	-	-
Total	3	1	-	2

3 | Vulnerability Summary

3.1 Overview

Click on an issue to jump to it, or scroll down to see them all.

~~H-1~~ Revisited Total Asset Calculation in SympieStaking

~~M-1~~ Revisited Logic of SympieStaking::transferCollateralToVaultDelegator()

L-1 Potential Risks Associated with Centralization

3.2 Security Level Reference

In web3 smart contract audits, vulnerabilities are typically classified into different severity levels based on the potential impact they can have on the security and functionality of the contract. Here are the definitions for critical-severity, high-severity, medium-severity, and low-severity vulnerabilities:

Severity	Description
C-X (Critical)	A severe security flaw with immediate and significant negative consequences. It poses high risks, such as unauthorized access, financial losses, or complete disruption of functionality. Requires immediate attention and remediation.
H-X (High)	Significant security issues that can lead to substantial risks. Although not as severe as critical vulnerabilities, they can still result in unauthorized access, manipulation of contract state, or financial losses. Prompt remediation is necessary.
M-X (Medium)	Moderately impactful security weaknesses that require attention and remediation. They may lead to limited unauthorized access, minor financial losses, or potential disruptions to functionality.
L-X (Low)	Minor security issues with limited impact. While they may not pose significant risks, it is still recommended to address them to maintain a robust and secure smart contract.
I-X (Informational)	Warnings and things to keep in mind when operating the protocol. No immediate action required.
U-X (Undetermined)	Identified security flaw requiring further investigation. Severity and impact need to be determined. Additional assessment and analysis are necessary.

3.3 Vulnerability Details

[H-1] Revisited Total Asset Calculation in SympieStaking

Target	Category	IMPACT	LIKELIHOOD	STATUS
SympieStaking.sol	Business Logic	High	High	Addressed

The `SympieStaking` contract is a core component for user interactions within the Symbiotic protocol. It enables users to deposit supported assets, which are then converted into corresponding collateral tokens. The `getAssetDistributionData()` function is designed to monitor the distribution of these collateral tokens across the `SympieStaking` contract, `VaultDelegator`, and `Vaults`, which is critical for determining the accurate exchange rate between `MLRT` and `LST`.

Upon review, a critical flaw is identified in the `getAssetDistributionData()` function. The function erroneously retrieves the amount of the original assets instead of the collateral tokens from the VaultDelegator contract (line 97). This mistake results in an inaccurate total asset calculation, potentially compromising the precision of the `MLRT` to `LST` exchange rate.

SympieStaking::getAssetDistributionData()

```
79  function getAssetDistributionData(
80      address asset
81  )
82      public
83      view
84      onlySupportedAsset(asset)
85      returns (
86          uint256 assetLyingInDepositPool,
87          uint256 assetLyingInVDC,
88          uint256 assetStakedInVaults
89      )
90  {
91      assetLyingInDepositPool = TransferHelper.balanceOf(
92          sympieConfig.assetCollateral(asset),
93          address(this)
94      );

96      assetLyingInVDC += TransferHelper.balanceOf(
97          asset,
98          sympieConfig.getContract(SympieConstants.VAULT_DELEGATOR)
99      );
100     ...
101 }
```

Remediation The implementation of the `getAssetDistributionData()` function need be corrected

to ensure it accurately retrieves the balance of collateral tokens rather than the original asset amounts.

[M-1] Revisited Logic of SympieStaking::transferCollateralToVaultDelegator()

Target	Category	IMPACT	LIKELIHOOD	STATUS
SympieStaking.sol	Business Logic	Medium	Medium	Addressed

The `transferCollateralToVaultDelegator()` function is designed for privileged accounts to transfer the `Symbiotic` collateral tokens deposited by users into the `VaultDelegator` contract. However, during our review, we identify a critical issue: it transfers the assets instead of the corresponding `Symbiotic` collateral tokens into the `VaultDelegator` contract (line 203).

This issue arises from a misunderstanding of the system's flow: when users deposit assets into the `SympieStaking` contract, these assets are immediately converted into collateral tokens via the `Symbiotic` protocol. As a result, there should be no original assets remaining in the `SympieStaking` contract and only collateral tokens should be present.

SympieStaking::transferCollateralToVaultDelegator()

```
198 function transferCollateralToVaultDelegator(
199     address asset,
200     uint256 amount
201 ) external nonReentrant onlyAllowedBot onlySupportedAsset(asset) {
202     TransferHelper.safeTransferToken(
203         asset,
204         sympieConfig.getContract(SympieConstants.VAULT_DELEGATOR),
205         amount
206     );
207 }
```

Remediation Correctly transfer collateral tokens to the `VaultDelegator` contract in the `transferCo-llateralToVaultDelegator()` function.

[L-1] Potential Risks Associated with Centralization

Target	Category	IMPACT	LIKELIHOOD	STATUS
Multiple Contracts	Security	Low	Low	Acknowledged

In the `Sympie` protocol, the existence of a series of privileged accounts introduces centralization risks, as they hold significant control and authority over critical operations governing the protocol. In the following, we show the representative function potentially affected by the privileges associated with the privileged accounts.

Example Privileged Operations in Sympie

```
93  function updateMLRTPrice(
94      address asset,
95      uint256 newExchangeRate
96  ) external onlyOracleAdmin {
97      address mLRTReceipt = sympieConfig.mLRTReceiptByAsset(asset);

99      _checkNewRate(mLRTReceipt, newExchangeRate);

101     IMLRT(mLRTReceipt).updateExchangeRateToLST(newExchangeRate);

103     emit ExchangeRateUpdate(asset, mLRTReceipt, newExchangeRate);
104 }

106 function updatePriceAdapterFor(
107     address asset,
108     address priceAdapter
109 ) external onlyOracleAdmin onlySupportedAsset(asset) {
110     UtilLib.checkNonZeroAddress(priceAdapter);
111     assetPriceOracle[asset] = priceAdapter;

113     emit AssetPriceAdapterUpdate(asset, priceAdapter);
114 }
```

Remediation To mitigate the identified issue, it is recommended to introduce multi-sig mechanism to undertake the role of the privileged accounts. Moreover, it is advisable to implement timelocks to govern all modifications to the privileged operations.

Response By Team This issue has been confirmed by the team.

4 Appendix

4.1 About AstraSec

AstraSec is a blockchain security company that serves to provide high-quality auditing services for blockchain-based protocols. With a team of blockchain specialists, AstraSec maintains a strong commitment to excellence and client satisfaction. The audit team members have extensive audit experience for various famous DeFi projects. AstraSec's comprehensive approach and deep blockchain understanding make it a trusted partner for the clients.

4.2 Disclaimer

The information provided in this audit report is for reference only and does not constitute any legal, financial, or investment advice. Any views, suggestions, or conclusions in the audit report are based on the limited information and conditions obtained during the audit process and may be subject to unknown risks and uncertainties. While we make every effort to ensure the accuracy and completeness of the audit report, we are not responsible for any errors or omissions in the report.

We recommend users to carefully consider the information in the audit report based on their own independent judgment and professional advice before making any decisions. We are not responsible for the consequences of the use of the audit report, including but not limited to any losses or damages resulting from reliance on the audit report.

This audit report is for reference only and should not be considered a substitute for legal documents or contracts.

4.3 Contact

Phone	+86 176 2267 4194
Email	contact@astrasec.ai
Twitter	https://twitter.com/AstraSecAI