Babypie

Security Audit Report

February 7, 2025

Contents

1 Introduction

1.1 About Babypie

Babypie is a top-tier SubDAO developed by Magpie that concentrates on liquid staking services for BTC using Babylon. As a liquid staking platform for Bitcoin, Babypie allows users to stake their Bitcoin as mBTC. Created by Babypie, mBTC is a liquid staked version of BTC, enabling users to earn rewards from Bitcoin staking without any required lockup period and providing passive income opportunities across DeFi.

1.2 Audit Scope

First Audit Scope

The following source code was reviewed during the audit:

- https://github.com/magpiexyz/babypie/tree/version2
- Commit ID: 0ff00ac

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

- https://github.com/magpiexyz/babypie/tree/version2
- Commit ID: a238b46

Second Audit Scope

The following source code was reviewed during the audit:

- https://github.com/magpiexyz/babypie/pull/7
- Commit ID: 6cbb0cf

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

- https://github.com/magpiexyz/babypie/pull/7
- Commit ID: df39300

Third Audit Scope

The following source code was reviewed during the audit:

- https://github.com/magpiexyz/babypie/pull/21
- Commit ID: 46a9528

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

- https://github.com/magpiexyz/babypie
- Commit ID: 78f0e52

Fourth Audit Scope

The following source code was reviewed during the audit:

- https://github.com/magpiexyz/babypie/pull/26
- Commit ID: 6e6b412

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

- https://github.com/magpiexyz/babypie
- Commit ID: 650e9b8

Fifth Audit Scope

The following source code was reviewed during the audit:

- https://github.com/magpiexyz/babypie/pull/29
- Commit ID: fe0bc3e

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

- https://github.com/magpiexyz/babypie
- Commit ID: debea50

Sixth Audit Scope

The following source code was reviewed during the audit:

- https://github.com/magpiexyz/babypie/pull/33
- Commit ID: 2bd8c04

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

- https://github.com/magpiexyz/babypie
- Commit ID: d625eba

1.3 Changelog

Version	Date
First Audit	June 30, 2024
Second Audit	August 28, 2024
Third Audit	November 25, 2024
Fourth Audit	December 19, 2024
Fifth Audit	January 10, 2025
Sixth Audit	February 3, 2025

2 | Overall Assessment

This report has been compiled to identify issues and vulnerabilities within the `Babypie` 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	-	-	-	-
Medium	2	1	-	1
Low	2	-	-	2
Informational	1	-	-	1
Undetermined	-	-	-	-

3 | Vulnerability Summary

3.1 Overview

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

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

[M-1] Potential Front-Running Risk in BabypieManager::initiateReceiptMint()

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

The `BabypieManager::initiateReceiptMint()` function allows users to initiate the minting of receipt tokens, also including setting a `referrer` parameter. This `referrer` parameter is necessary for off-chain calculations using a subgraph, as it needs to be emitted after a successful mint. However, there are no restrictions on who can call this function, allowing anyone to execute it. This lack of restriction opens up the potential for a malicious actor to front-run a legitimate user and change the `referrer` parameter to an address that benefits them instead.

BabypieManager::initiateReceiptMint()

```
97    function initiateReceiptMint(string calldata btcTxnHash, string calldata
          userAddress, address referrer) public payable {
98          if(btcTxnInfo[btcTxnHash].isMinted)
99              revert alreadyMintedForThisTxn();
100         if(msg.value < txnFeeValue)
101             revert NotEnoughTxnFeeSent();

103         string[] memory args = new string[](3);
104         args[0] = btcTxnHash;
105         args[1] = magpieCustodianWallet;
106         args[2] = userAddress;
107         ITransactionDataProvider(txnDataProvider).getTxnDataAndMint(args,
                referrer);
108     }
```

Remediation It is recommended to allow the `referrer` parameter to be set only when `msg.sender` equals the `userEVMAddress`.

[M-2] Potential Risks Associated with Centralization

Target	Category	IMPACT	LIKELIHOOD	STATUS
Multiple Contracts	Security	Medium	Medium	Acknowledged

In the `Babypie` protocol, the existence of a privileged `owner` account introduces centralization risks, as it holds significant control and authority over critical operations governing the protocol. In the following, we show the representative functions potentially affected by the privileges associated with the privileged account.

Example Privileged Operations in `Babypie` protocol

```
151 function setmBTC(address _mBTC) external onlyOwner {
152     mBTC = _mBTC;
153 }

155 function setChainlinkFunctions(address _verificationProvider, address
        _txnDataProvider) external onlyOwner {
156     verificationProvider = _verificationProvider;
157     txnDataProvider = _txnDataProvider;
158 }

160 function setMagpieCustodianWallet(string calldata _walletAddress) external
        onlyOwner {
161     magpieCustodianWallet = _walletAddress;
162 }

164 function setTxnFee(uint256 feeValue) external onlyOwner {
165     txnFeeValue = feeValue;
166 }
```

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. The multi-sig mechanism will be used to mitigate this issue.

[L-1] Suggested Adding Pause Support for Functions in BabypieManager

Target	Category	IMPACT	LIKELIHOOD	STATUS
BabypieManager.sol	Business Logic	Low	Low	Addressed

The `BabypieManager` contract inherits the `ReentrancyGuardUpgradeable` and `PausableUpgradeable` contracts but does not utilize the functionalities provided by these two contracts. It is recommended to add two privileged functions, `pause()` and `unpause()`, to the current implementation, allowing the privileged owner account to pause/unpause the `BabypieManager` contract. Additionally, apply the `whenNotPaused()` modifier to the public functions `initiateReceiptMint()` and `initiateUserAddressUpdate()`, ensuring these functions can only be called when the contract is in the unpaused state.

BabypieManager.sol

```
18  contract BabypieManager is
19    Initializable,
20    OwnableUpgradeable,
21    ReentrancyGuardUpgradeable,
22    PausableUpgradeable
23  {
24    ...
25    /* ====================Chainlink Function initiator public functions
          ================*/

27    function initiateReceiptMint(string calldata btcTxnHash, string calldata
          userAddress, address referrer) public payable {
28        if(btcTxnInfo[btcTxnHash].isMinted)
29            revert alreadyMintedForThisTxn();
30        if(msg.value < txnFeeValue)
31            revert NotEnoughTxnFeeSent();

33        string[] memory args = new string[](3);
34        args[0] = btcTxnHash;
35        args[1] = magpieCustodianWallet;
36        args[2] = userAddress;
37        ITransactionDataProvider(txnDataProvider).getTxnDataAndMint(args, referrer
              );
38    }

40    function initiateUserAddressUpdate(string calldata _userBTCAddress, string
          calldata _evmAddress, string calldata _userSignature) public payable {
41        if(userInfo[_userBTCAddress].evmAddress != address(0))
42            revert EVMAddressAlreadyMapped();
43        if(msg.value < txnFeeValue)
44            revert NotEnoughTxnFeeSent();
45        string[] memory args = new string[](3);
```

```
        args[0] = _userBTCAddress;
        args[1] = _evmAddress;
        args[2] = _userSignature;
        IVerificationProviderFunction(verificationProvider).
            verifySignatureAndUpdateAddress(args);
    }
    ...
}
```

Remediation Add a pause/unpause mechanism to the `BabypieManager` contract and ensure that key functions can only be executed when the contract is in the unpaused state.

[L-2] Logical Inconsistencies in setBabyPieCustodianWalletInfo()

Target	Category	IMPACT	LIKELIHOOD	STATUS
BabypieManager.sol	Business Logic	Low	Low	Addressed

The `setBabyPieCustodianWalletInfo()` function in the `BabypieManager` contract allows the privileged `owner` account to update or set the `BabyPie` custodian wallet information at a specified index. It enables the contract `owner` to set or modify the wallet's address, activation status (`isActive`), and privacy status (`isPrivat`). Upon reviewing the contract's implementation, we identified a logical flaw in the function. Specifically, if the wallet's `isPrivate` status changes (e.g., from false to true), the function lacks the necessary logic to update or remove the user address associated with that wallet address, which could lead to inconsistent data state. Additionally, the function does not perform a uniqueness check on the `_walletAddress` input parameter. It does not verify whether `_walletAddress` already exists in other `BabypieCustodianWalletInfo` entries, which could result in different custodian wallet entries using the same address, causing data inconsistency or potential security issues.

BabypieManager::setBabyPieCustodianWalletInfo()

```
function setBabyPieCustodianWalletInfo(uint256 _index, string calldata
    _walletAddress, bool _isActive, bool _isPrivate) external onlyOwner {
    if(bytes(_walletAddress).length == 0)
        revert AddressZero();
    if (_index >= babypieCustodianWallets.length  _index < 0)
        revert InvalidIndex();

    BabypieCustodianWalletInfo storage babyPieCustodianWallet =
        babypieCustodianWallets[_index];
    babyPieCustodianWallet.babypieCustodianWalletAddress = _walletAddress;
    babyPieCustodianWallet.isActive = _isActive;
    babyPieCustodianWallet.isPrivate = _isPrivate;
```

```
318     emit BabyPieCustodianWalletInfoSet(_index, _walletAddress, _isActive,
            _isPrivate);
319 }
```

Remediation Include logic to update or delete user address associations when the `isPrivate` status is changed, ensuring consistent data management across private and public wallet states. Additionally, implement a uniqueness check for the `_walletAddress` to ensure it doesn't already exist in another `BabypieCustodianWalletInfo` entry. An example revision is shown as follows:

BabypieManager::setBabyPieCustodianWalletInfo()

```
307 function setBabyPieCustodianWalletInfo(uint256 _index, string calldata
        _walletAddress, string calldata _userAddress, bool _isActive, bool _isPrivate
        ) external onlyOwner {
308     if(bytes(_walletAddress).length == 0)
309         revert AddressZero();
310     if (_index >= babypieCustodianWallets.length)
311         revert InvalidIndex();

313     for (uint256 i = 0; i < babypieCustodianWallets.length; i++) {
314         if (i != _index && keccak256(bytes(babypieCustodianWallets[i].
                babypieCustodianWalletAddress)) ==  keccak256(bytes(_walletAddress)))
                 {
315             revert AddressAlreadyExists();
316         }
317     }

319     BabypieCustodianWalletInfo storage babyPieCustodianWallet =
            babypieCustodianWallets[_index];

321     if (babyPieCustodianWallet.isPrivate) {
322         if (!_isPrivate  keccak256(bytes(babypieCustodianWallets[_index].
                babypieCustodianWalletAddress)) !=  keccak256(bytes(_walletAddress)))
                 {
323             delete babypieCustodianWalletForUser[babypieCustodianWallets[_index
                    ].babypieCustodianWalletAddress];
324         }

326         if (_isPrivate) {
327             babypieCustodianWalletForUser[_walletAddress] = _userAddress;
328         }
329     }

331     if (!babyPieCustodianWallet.isPrivate && _isPrivate) {
332         babypieCustodianWalletForUser[_walletAddress] = _userAddress;
333     }
```

```
335         babyPieCustodianWallet.babypieCustodianWalletAddress = _walletAddress;
336         babyPieCustodianWallet.isActive = _isActive;
337         babyPieCustodianWallet.isPrivate = _isPrivate;

339         emit BabyPieCustodianWalletInfoSet(_index, _walletAddress, _isActive,
                _isPrivate);
340     }
```

[I-1] Improved Sanity Checks in BabypieManager::initiateReceiptMint()

Target	Category	IMPACT	LIKELIHOOD	STATUS
BabypieManager.sol	Business Logic	N/A	N/A	Addressed

After successfully transferring `Bitcoin` to the `Magpie` custodian wallet, the user calls the `BabypieManager::initiateReceiptMint()` function to initiate the minting of the receipt token to the user. When examining its implementation logic, we found that it could benefit from adding additional sanity checks.

To elaborate, we show the related code snippet below. Specifically, the current implementation does not check whether the corresponding `EVM` address mapping is set for the input `BTC` address. If `userInfo[userAddress].evmAddress == address(0)`, the function execution becomes meaningless and results in a waste of gas.

BabypieManager::initiateReceiptMint()

```
97      function initiateReceiptMint(string calldata btcTxnHash, string calldata
            userAddress, address referrer) public payable {
98          if(btcTxnInfo[btcTxnHash].isMinted)
99              revert alreadyMintedForThisTxn();
100         if(msg.value < txnFeeValue)
101             revert NotEnoughTxnFeeSent();

103         string[] memory args = new string[](3);
104         args[0] = btcTxnHash;
105         args[1] = magpieCustodianWallet;
106         args[2] = userAddress;
107         ITransactionDataProvider(txnDataProvider).getTxnDataAndMint(args,
                referrer);
108     }
```

Remediation Perform the necessary sanity check on the input `BTC` address, and proceed with subsequent operations only if the corresponding `EVM` address mapping is set.

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	+86 156 0639 2692
Email	contact@astrasec.ai
Twitter	https://twitter.com/AstraSecAI