ChainLight

Biconomy

# Biconomy Sponsorship Paymaster Security Audit

: Sponsorship Paymaster

---

Oct 23, 2024

Revision 1.1

ChainLight@Theori

# Table of Contents

# Executive Summary

Starting on Oct 1st, 2024, ChainLight of Theori audited the smart contract of Biconomy Sponsorship Paymaster for one week. In the audit, we primarily considered the issues/impacts listed below.

- Theft of funds
- Reputation loss of the paymaster's due to ERC-4337 standard violation
- PostOp failure possibility
- Use of funds from other Paymasters

As a result, we identified issues as listed below.

- Total: 6
- Critical: 1 (Theft of funds)
- High: 2 (Reputation loss of the paymaster's due to ERC-4337 standard violation, PostOp failure possibility)
- Medium: 1 (Use of funds from other Paymasters)
- Low: 1
- Informational: 1

# Audit Overview

## Scope

| | |
|---|---|
| **Name** | Biconomy Sponsorship Paymaster Security Audit |
| **Target / Version** | • Git Repository ( `sponsorship/BiconomySponsorshipPaymaster.sol` ): commit `acc0dca29fc53b21aca7fe24da85d664a11da46a` |
| **Application Type** | Smart contracts |
| **Lang. / Platforms** | Smart contracts [Solidity] |

## Code Revision

N/A

## Severity Categories

| Severity | Description |
|---|---|
| **Critical** | The attack cost is low (not requiring much time or effort to succeed in the actual attack), and the vulnerability causes a high-impact issue. (e.g., Effect on service availability, Attacker taking financial gain) |
| **High** | An attacker can succeed in an attack which clearly causes problems in the service's operation. Even when the attack cost is high, the severity of the issue is considered "high" if the impact of the attack is remarkably high. |
| **Medium** | An attacker may perform an unintended action in the service, and the action may impact service operation. However, there are some restrictions for the actual attack to succeed. |
| **Low** | An attacker can perform an unintended action in the service, but the action does not cause significant impact or the success rate of the attack is remarkably low. |
| **Informational** | Any informational findings that do not directly impact the user or the protocol. |
| **Note** | Neutral information about the target that is not directly related to the project's safety and security. |

## Status Categories

| Status | Description |
| --- | --- |
| Reported | ChainLight reported the issue to the client. |
| WIP | The client is working on the patch. |
| Patched | The client fully resolved the issue by patching the root cause. |
| Mitigated | The client resolved the issue by reducing the risk to an acceptable level by introducing mitigations. |
| Acknowledged | The client acknowledged the potential risk, but they will resolve it later. |
| Won't Fix | The client acknowledged the potential risk, but they decided to accept the risk. |

## Finding Breakdown by Severity

| Category | Count | Findings |
| --- | --- | --- |
| Critical | 1 | • `BICONOMYSPONSOR-001` |
| High | 2 | • `BICONOMYSPONSOR-004`<br>• `BICONOMYSPONSOR-005` |
| Medium | 1 | • `BICONOMYSPONSOR-002` |
| Low | 1 | • `BICONOMYSPONSOR-003` |
| Informational | 1 | • `BICONOMYSPONSOR-006` |
| Note | 0 | • N/A |

# Findings

## Summary

| # | ID | Title | Severity | Status |
|---|---|---|---|---|
| 1 | BICONOMYSPONSOR-001 | `BiconomySponsorshipPaymaster` must deduct `paymasterIdBalances` during the verification phase | Critical | Patched |
| 2 | BICONOMYSPONSOR-002 | `priceMarkup` must be greater than or equal to `PRICE_DENOMINATOR` | Medium | Patched |
| 3 | BICONOMYSPONSOR-003 | Mitigating centralized risk by transitioning signature verification from `verifyingSigner` to `paymasterId` | Low | Won't Fix |
| 4 | BICONOMYSPONSOR-004 | `validatePaymasterUserOp()` must validate `postOpGasLimit` to ensure the execution of postOp | High | Patched |
| 5 | BICONOMYSPONSOR-005 | BiconomySponsorshipPaymaster is vulnerable to reputation decay attacks | High | Patched |
| 6 | BICONOMYSPONSOR-006 | Minor Suggestions | Informational | Patched |

## #1 BICONOMYSPONSOR-001 BiconomySponsorshipPaymaster must deduct paymasterIdBalances during the verification phase

| ID | Summary | Severity |
|---|---|---|
| BICONOMYSPONSOR-001 | If the total gas usage required by multiple UserOperations in a batch exceeds the value of `paymasterIdBalances[paymasterId]`, an underflow may occur in `paymasterIdBalances[paymasterId]` during the `postOp` phase. | Critical |

### Description

In `BiconomySponsorshipPaymaster._validatePaymasterUserOp()`, the gas amount required for each `UserOperation`, called `effectiveCost`, is checked to ensure that it does not exceed `paymasterIdBalances[paymasterId]`. This gas amount is deducted in the `_postOp()` function. However, this approach does not consider the case where multiple UserOperations reference a single `paymasterId` simultaneously. When the total `effectiveCost` from multiple UserOperations exceeds `paymasterIdBalances[paymasterId]`, each UserOperation's `effectiveCost` is individually validated in `_validatePaymasterUserOp()` without errors, but an underflow occurs in `_postOp()` when deducting from `paymasterIdBalances[paymasterId]`. This situation allows the `paymasterId` address to withdraw all the ETH funds deposited by the paymaster via the `withdrawTo()` function. Thus, the balance of `paymasterIdBalances` should be deducted in `_validatePaymasterUserOp()` rather than in `_postOp()`.

Additionally, when calculating `effectiveCost` in `_validatePaymasterUserOp()`, the value of `unaccountedGas` must be considered. The value of `unaccountedGas * actualUserOpFeePerGas` is not included when calculating `effectiveCost`, but it is included in the `adjustedGasCost` calculation in `_postOp()`. This discrepancy may lead to a situation where `effectiveCost > adjustedGasCost`, resulting in an underflow in `paymasterIdBalances[paymasterId] -= adjustedGasCost`. Furthermore, even if `unaccountedGas` is accounted for when calculating `effectiveCost`, it can increase during the execution phase of the UserOperation due to a call to `setUnaccountedGas()`. To prevent such scenarios, `unaccountedGas` should be included in the `context` of

`_validatePaymasterUserOp()` and decoded from the `context` in `_postOp()` for use in gas calculations.

## Impact

**Critical**

If an underflow occurs in `paymasterIdBalances[paymasterId]`, the address associated with the `paymasterId` can steal all the ETH deposited by other paymasters.

## Recommendation

Implement all of the following recommendations:

1. Decrease the balance of `paymasterIdBalances[paymasterId]` in `_validatePaymasterUserOp()` instead of in `_postOp()`.
2. Modify the calculation of `effectiveCost` to `(requiredPreFund + unaccountedGas * userOp.unpackMaxFeePerGas()) * priceMarkup / PRICE_DENOMINATOR`.
3. Include `effectiveCost` and `unaccountedGas` in the `context` in `_validatePaymasterUserOp()` and use this in `_postOp()` to calculate the remaining gas. And refund the excess amount to `paymasterId`.

## Remediation

**Patched**

The issue has been resolved as recommended.

## #2 BICONOMYSPONSOR-002 priceMarkup must be greater than or equal to PRICE_DENOMINATOR

| ID | Summary | Severity |
| --- | --- | --- |
| BICONOMYSPONSOR-002 | If priceMarkup is smaller than PRICE_DENOMINATOR, a single paymasterId address can infringe upon and use the ETH deposited by other paymasterId addresses. | Medium |

### Description

When priceMarkup is smaller than PRICE_DENOMINATOR, the amount deducted from paymasterIdBalances[paymasterId] can be less than the gas consumption of the UserOperation. If priceMarkup is half the value of PRICE_DENOMINATOR, a specific paymasterId could sponsor twice the amount of gas usage for UserOperations compared to the ETH it deposited. This excess sponsorship can be drawn from the ETH deposited by other paymasterId addresses. Therefore, this discount scenario is not appropriate in the current singleton structure where multiple paymasterId addresses deposit ETH into a single paymaster.

### Impact

**Medium**

If priceMarkup is smaller than PRICE_DENOMINATOR, a single paymasterId can infringe upon and use the ETH deposited by other paymasterId addresses to sponsor operations.

### Recommendation

It is recommended to validate that priceMarkup is greater than or equal to PRICE_DENOMINATOR.

### Remediation

**Patched**

The issue has been resolved as recommended.

## #3 `BICONOMYSPONSOR-003` Mitigating centralized risk by transitioning signature verification from `verifyingSigner` to `paymasterId`

| ID | Summary | Severity |
| --- | --- | --- |
| `BICONOMYSPONSOR-003` | Centralization risks can be mitigated by performing signature verification through the `paymasterId` address instead of the `verifyingSigner`. | Low |

### Description

The current structure requires `paymasterId` addresses, which deposit ETH into the paymaster, to trust a single `verifyingSigner`. If the `verifyingSigner` is malicious, it could generate a UserOperation that pays an unusually high fee to the bundler (by setting a high gas price), allowing it to steal ETH deposited by the `paymasterId` addresses. To eliminate this centralized risk, it is recommended to modify the structure so that signatures are received and verified directly from each `paymasterId` instead of the `verifyingSigner`.

### Impact

**Low**

If the `verifyingSigner` is malicious, it could steal ETH deposited by `paymasterId` addresses into the paymaster.

### Recommendation

It is recommended to perform signature verification through the `paymasterId` address rather than the `verifyingSigner`.

### Remediation

**Won't Fix**

Biconomy team decided to retain the `verifyingSigner` so that the `paymasterId` doesn't need to handle any signature-related processing.

## #4 BICONOMYSPONSOR-004 validatePaymasterUserOp() must validate postOpGasLimit to ensure the execution of postOp

| ID | Summary | Severity |
| --- | --- | --- |
| BICONOMYSPONSOR-004 | _validatePaymasterUserOp() in BiconomySponsorshipPaymaster must ensure that userOp.unpackPostOpGasLimit() is greater than the actual cost of executing the post-operation in order to guarantee the execution of postOp. | High |

### Description

If the postOpGasLimit in userOperation is set too low, the postOp may fail due to an out-of-gas error. In this case, although the user's operation will also be reverted, the transaction will still be processed, leading to a decrease in the Paymaster's eth balance in the EntryPoint. Meanwhile, since the post-operation includes the logic to reduce paymasterIdBalances[paymasterId] and collect fees, a failure in postOp will result in the reduction of the Paymaster's eth balance without properly decreasing the paymasterIdBalances[paymasterId]. This allows a specific paymasterId to sponsor more funds than it has deposited.

### Impact

**High**

When the postOpGasLimit of userOperation is set too low, postOp may fail due to an out-of-gas error. In such cases, while the Paymaster's actual eth balance decreases, the fee collection and paymasterIdBalances[paymasterId] adjustment are not performed, potentially leading to discrepancies. However, the attack can be mitigated off-chain by having the verifyingSigner validate the postOpGasLimit.

### Recommendation

It is recommended to add require(refundPostopCost < userOp.unpackPostOpGasLimit(), "postOpGasLimit too low") to BiconomySponsorshipPaymaster._validatePaymasterUserOp().

## Remediation

**Patched**

The issue has been resolved as recommended.

## #5 `BICONOMYSPONSOR-005` BiconomySponsorshipPaymaster is vulnerable to reputation decay attacks

| ID | Summary | Severity |
| --- | --- | --- |
| `BICONOMYSPONSOR-005` | A malicious `paymasterId` can reduce the reputation of the paymaster, leading to a DoS (Denial of Service) attack. | High |

### Description

The `BiconomySponsorshipPaymaster` is a singleton structure where multiple Dapps and Wallet Clients deposit ETH into a single paymaster, represented by a `paymasterId`, and use the deposited amount to sponsor their users. Because multiple services use a single paymaster, an attack that damages the paymaster's reputation could also affect other services that use the same paymaster.

An attack on the reputation of a specific `paymasterId` can be executed by generating a large number of UserOperations. The bundler increases the `opsSeen` count of the paymaster, account, and factory referenced by each UserOperation when they are added to the mempool. By generating and submitting multiple UserOperations linked to the paymaster and then calling `withdrawTo()` to withdraw all funds before the UserOperations are included on-chain, the attacker can invalidate all UserOperations that reference the specific `paymasterId`. While these UserOperations are eventually removed from the mempool during the second verification phase when the bundler creates a bundle, the `opsSeen` count does not decrease. The attacker can continue to increase the paymaster's `opsSeen` count by repeating the `depositFor()` and withdrawal process, reducing the paymaster's reputation. If the paymaster's reputation falls to the `THROTTLED` or `BAN` level, other `paymasterId`s using the same paymaster will also be unable to use sponsorship transactions.

### Impact

**High**

If a specific `paymasterId` calls `withdrawTo()`, all UserOperations referencing that `paymasterId` become invalid. When a large number of such UserOperations exist in the mempool simultaneously, the bundler reduces the paymaster's reputation, leading to `THROTTLED` or `BAN`

status. If the paymaster is banned, the bundler rejects all UserOperations that reference the paymaster, preventing other `paymasterId`s from using gas sponsorship through that paymaster.

### Recommendation

Both off-chain and on-chain measures should be applied.

(Off-chain) The `verifyingSigner` should limit the number of UserOperations that reference the same `paymasterId` at the same time. This will prevent too many UserOperations from becoming invalid in the mempool at once. Additionally, it is recommended to verify whether the `maxFeePerGas` and `maxPriorityFeePerGas` values are reasonable to prevent situations where the bundler does not process UserOperations.

(On-chain) To prevent the invalidation of UserOperations referencing a `paymasterId` due to repeated `withdrawTo()` calls, it is recommended to add a delay to the `withdrawTo()` function and set a minimum deposit amount for the `depositFor()` function.

### Remediation

**Patched**

The issue has been resolved as recommended.

## #6 BICONOMYSPONSOR-006 Minor Suggestions

| ID | Summary | Severity |
| --- | --- | --- |
| BICONOMYSPONSOR-006 | The description includes multiple suggestions for preventing incorrect settings caused by operational mistakes, mitigating potential issues, improving code maturity and readability, and other minor issues. | Informational |

### Description

- `withdrawTo()` should revert when the `amount` is `0`. This can prevent unnecessary events from being emitted.
- Typo in the comment for `setUnaccountedGas()`. It should be `unaccountedGas` instead of `unaccountedGasOverhead`.

### Impact

**Informational**

### Recommendation

Consider applying the suggestions in the description above.

### Remediation

**Patched**

All suggestions are applied as recommended.

## Revision History

| Version | Date | Description |
| --- | --- | --- |
| **1.0** | Oct 15, 2024 | Initial version |
| **1.1** | Oct 23, 2024 | `BICONOMYSPONSOR-005` status change |

Theori, Inc. (“We”) is acting solely for the client and is not responsible to any other party. Deliverables are valid for and should be used solely in connection with the purpose for which they were prepared as set out in our engagement agreement. You should not refer to or use our name or advice for any other purpose. The information (where appropriate) has not been verified. No representation or warranty is given as to accuracy, completeness or correctness of information in the Deliverables, any document, or any other information made available. Deliverables are for the internal use of the client and may not be used or relied upon by any person or entity other than the client. Deliverables are confidential and are not to be provided, without our authorization (preferably written), to entities or representatives of entities (including employees) that are not the client, including affiliates or representatives of affiliates of the client.