CANTINA

Zyfi Permissionless multi-signer paymaster

Security Review

Cantina Managed review by:
Deadrosesxyz, Lead Security Researcher
Chris Smith, Security Researcher

August 7, 2024

Contents

1 Introduction **2**
- 1.1 About Cantina . . . 2
- 1.2 Disclaimer . . . 2
- 1.3 Risk assessment . . . 2
  - 1.3.1 Severity Classification . . . 2

2 Security Review Summary **3**

3 Findings **4**
- 3.1 High Risk . . . 4
  - 3.1.1 Attacker might repeatedly use `selfRevokeSigner` to steal gas refunds . . . 4
- 3.2 Low Risk . . . 4
  - 3.2.1 Signature Replay risk up to `expirationTime` and `maxNonce` . . . 4
- 3.3 Informational . . . 5
  - 3.3.1 Solidity variable naming best practices are not followed . . . 5
  - 3.3.2 Consistent use of `uint256` vs `uint` . . . 5
  - 3.3.3 Consistent conditional logic will make `updateRefund` easier to reason about . . . 5
  - 3.3.4 Floating pragma used . . . 5

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Zyfi is a Paymaster-as-a-Service that focuses on flexibility and ease of integration. It was created as a way to accelerate the adoption of paymasters in the zkSync ecosystem.

From Jun 27th to Jul 2nd the Cantina team conducted a review of permissionless-multisigner-paymaster on commit hash 29943aba and a change review by Deadrosesxyz on Aug 2nd on commit hash 86d0a6b1. The team identified a total of **6** issues in the following risk categories:

- Critical Risk: 0
- High Risk: 1
- Medium Risk: 0
- Low Risk: 1
- Gas Optimizations: 0
- Informational: 4

3 Findings

3.1 High Risk

3.1.1 Attacker might repeatedly use `selfRevokeSigner` to steal gas refunds

Severity: High Risk

Context: PermissionlessPaymaster.sol#L377

Description: Currently, the `selfRevokeSigner` sets `previousManager` to the signer's currently set manager.

```
function selfRevokeSigner() public {
    previousManager = managers[msg.sender];
    managers[msg.sender] = address(0);
    emit SignerRevoked(previousManager, msg.sender);
}
```

In case `updateRefund` hasn't been called since the last refund, this would lead to distributing the refund to the just revoked manager (instead of the actual `previousManager`).

An attacker can utilize this and back-run all transactions that utilize the Paymaster and ultimately steal all gas refunds.

Recommendation: Do not change `previousManager` within `selfRevokeSigner`.

Zyfi: Fixed in PR 1.

Cantina Managed: Fix looks good. `previousManager` is no longer changed within `selfRevokeSigner`.

3.2 Low Risk

3.2.1 Signature Replay risk up to `expirationTime` and `maxNonce`

Severity: Low Risk

Context: PermissionlessPaymaster.sol#L306-L317

Description: Signers need to ensure `maxNonce` and `expirationTime` are set and signed with appropriate values. If `maxNonce` is set high, then any user would be able reuse a signature from a signer to get the manager to pay for multiple transactions as long as the transaction data are the same (`_from`, `_to`, `_maxFeePerGas`, `_gasLimit`, `_markupPercent`) and the signature has not expired (`block.timestamp >= _expirationTime`). Due to the flexibility of `maxNonce`, the difference between the user's current nonce and the `maxNonce` should be viewed by signers and managers as the maximum number of times the manager is willing to pay for that transaction.

Recommendation: As has been discussed here and in the previous Zyfi paymaster audit, this is an intentional design trade-off. In order to provide flexibility to Dapps and users and to avoid the additional gas cost of an internal nonce system that would completely eliminate signature replay issues, there is some accepted risk.

The appropriate mitigation in this case is for signers to always use short `expirationTime`s and as low as possible `maxNonce` values. Managers should track and quickly respond if they identify signers using unacceptable `maxNonce` and `expirationTime` values as there is a chance funds could be drained. This will limit their exposure to one signature being used as authorization to pay for multiple transactions.

Zyfi: We acknowledge this design trade off. We wanted to provide as much as flexibility to Dapps. It is expected that signers would ensure this before signing. We will be collecting feedback from community and deploy a newer version of paymaster with no flexibility on nonce either by replacing `maxNonce` check to `currentNonce` or maintain internal nonce system if requested by the ecosystem.

Cantina Managed: This makes perfect sense. One additional note is to ensure that this design decision is well documented for Dapps.

3.3 Informational

3.3.1 Solidity variable naming best practices are not followed

Severity: Informational

Context: PermissionlessPaymaster.sol#L58

Description: Although, the variable `ZYFI_TREASURY` is not a constant, it is written in upper case. It is a Solidity best practice to only use upper case for variable naming when dealing with constants.

```
address public ZYFI_TREASURY;
```

Recommendation: Change the variable's name to `zyfi_treasury`.

Zyfi: Fixed in PR 2.

Cantina Managed: Fixed.

3.3.2 Consistent use of `uint256` vs `uint`

Severity: Informational

Context: Global scope

Description: In several places, the code uses `uint` as opposed to `uint256` [1, 2, 3, etc..]. While `uint` is an alias of `uint256`, it would be better to have it consistent across the code. Additionally, mixing uses were to apply to function signature encoding it could result in a bug (This is not the case currently, so this is an information/best practice note).

Recommendation: Update `uint` uses to `uint256` for consistency.

Zyfi: Fixed in PR 4.

Cantina Managed: Fix looks good, appears to cover all instances of `uint`.

3.3.3 Consistent conditional logic will make `updateRefund` easier to reason about

Severity: Informational

Context: PermissionlessPaymaster.sol#L58

Description: Reversing the conditionals in `updateRefund` makes the logic harder to reason about.

Recommendation: Have both `if` statements rely on `!withdraw` so there is less potential for confusion.

Zyfi: For better readability, fix in PR 3. Making it `isDeposit` would cost 5 additional gas and increase the complexity of the change, hence we decided to move forward with the current design.

Cantina Managed: Fix looks good and agree on the keeping it `!isWithdraw` over `isDeposit` for last minute change and gas reasons.

3.3.4 Floating pragma used

Severity: Informational

Context: PermissionlessPaymaster.sol#L24

Description: Currently, the contract uses a floating pragma, allowing the contract to be compiled with any `0.8.x` Solidity version higher than 0.8.18.

It is a security best practice to set the pragma to a specific version, in order to make sure the contract is not accidentally compiled to a version which breaks the contract's operability.

Recommendation: Set the pragma to a specific version.

Zyfi: Fixed in PR 7.

Cantina Managed: Fixed.