CANTINA

Biconomy Nexus PR212 & PR215

Security Review

Cantina Managed review by:
Chinmay Farkya, Associate Security Researcher

November 20, 2024

Contents

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

Biconomy is the world's most advanced ERC-4337 account abstraction infrastructure platform. Leverage our highly customisable transaction infra to build seamless dApps for every user.

On Nov 9th the Cantina team conducted a review of nexus on commit hash fa415c44.

The scope of the review was Biconomy's nexus PR 212 and PR 215.

The Cantina team reviewed Biconomy nexus changes holistically on commit hash 72499c9d and determined that all issues were resolved and no new issues were identified.

The team identified a total of **7** issues in the following risk categories:

- Critical Risk: 0
- High Risk: 0
- Medium Risk: 1
- Low Risk: 2
- Gas Optimizations: 1
- Informational: 3

3 Findings

3.1 Medium Risk

3.1.1 In `Nexus:checkERC7739Support()` only the last validator is checked to determine if the smart account supports ERC7739

Severity: Medium Risk

Context: Nexus.sol#L336-L350

Description: `isValidSignature()` → `checkERC7739Support()` flow in `Nexus.sol` is meant to determine if the smart account supports ERC7739. Nexus does so by looping over all the installed validators and returning true if any of the validators shows support for 7739 flows.

But the problem is that the logic only checks for one validator (i.e. the last entry in the sentinel list), This is because the logic uses a while loop starting from the sentinel entry but never increments the loop, so only one iteration is run and the loop ends.

Recommendation: Add `getNext()` method from SentinelList.sol#L54 to fetch the corresponding mapping value of an entry, which will behave as an increment operation inside the while loop. This will make the loop go over the complete list of validators.

Biconomy: Fixed in PR 216.

Cantina Managed: Fixed.

3.2 Low Risk

3.2.1 Wrong docs for `isValidSignatureWithSender()` in `K1Validator`

Severity: Low Risk

Context: K1Validator.sol#L158-L161

Description: The natspec says that the returned 4 bytes is the eip712 signature validation result (ie. either `0xffffffff` for invalid signature or `0x1626ba7e` for a valid signature).

But this is not always true. The `K1Validator` also inherits from `ERC7739Validator` and the `isValidSignatureWithSender()` → `_erc1271IsValidSignatureWithSender()` flow involves returning `bytes4(0x77390001)` for detection requests if any validator installed on the smart account supports `ERC7739`.

This means that `isValidSignatureWithSender()` might also end up returning `0x77390001`, which is not reflected in the natspec. This might confuse integration dapps and users who want to use this feature.

Recommendation: Add relevant natspec documentation for the mentioned case of this function possibly returning `0x77390001` in response to ERC7739 detection requests.

Biconomy: Fixed in PR 216.

Cantina Managed: Fixed.

3.2.2 `isValidSignatureWithSender()` also checks signature `s` value for `ERC7739` support detection calls which is unnecessary

Severity: Low Risk

Context: K1Validator.sol#L162-L179

Description: `K1Validator::isValidSignatureWithSender()` is used by the Nexus in two cases:

- When checking support for `erc7739` via `checkERC7739Support()` with `signature.length == 0` i.e. empty bytes.
- When doing actual signature validation from `isValidSignature()` → `isValidSignatureWithSender()`.

The `isValidSignatureWithSender()` function in `K1Validator` always applies checks on the `s` value even though its not needed in case 1. This could potentially result in the `s` value derived from the calldata to be an arbitrary value that creates problems. In any case, this check is unnecessary for the support detection requests as signature is expected to be empty bytes.

Recommendation: Move the entire `s` check logic to `ERC7739Validator :: _erc1271IsValidSignatureWithSender()` after the erc7739 checks logic ERC7739Validator.sol#L42-L48.

This saves gas for all case 1 calls and also prevents any potential problems if s value derived from calldata can come out to be a very large value.

Biconomy: Fixed in PR 216.

Cantina Managed: Fixed.

3.3 Gas Optimization

3.3.1 `isValidSignature()` in Nexus.sol can be optimized for cases when the call is not a `ERC7739` detection request

Severity: Gas Optimization

Context: Nexus.sol#L227-L240

Description: Currently `isValidSignature()` on Nexus calls into `checkERC7739Support()` for all calls. `checkERC7739Support()` returns false in two cases:

- When the call is not a `ERC7739` support detection request, but a normal `ERC1271` signature validation call (i.e. `signature.length != 0`).
- When none of the validators installed on the SA support `ERC7739`.

Specifically, case 1 can be optimized for gas. `checkERC7739Support()` does not need to be called if the call isn't a detection request.

Recommendation: Add this line to `isValidSignature()`:

```
if(signature.length == 0) {
    if(checkERC7739Support() return SUPPORTS_ERC7739;
}

/*
 * ...
 * Normal sig validation
 * ...
*/
```

And remove `If (signature.length == 0)` check from `checkERC7739Support()`. This will save gas used up in the unnecessary call to `checkERC7739Support()` for normal signature validation cases.

Biconomy: Fixed in PR 216.

Cantina Managed: Fixed.

3.4 Informational

3.4.1 Newer version of solady's ECDSA library available

Severity: Informational

Context: K1Validator.sol#L15

Description: Nexus uses solady ECDSA library version 0.0.246 right now, while newer versions are available.

Recommendation: Update the solady dependency to the latest version before deployment, to take advantage of new optimizations.

Biconomy: Fixed in PR 216.

Cantina Managed: Fixed.

3.4.2 Change in `ERC7739` magic number can break things when `ERC7739` standard gets updated

Severity: Informational

Context: Constants.sol#L49

Description: EIP7739 says that the magic number `0x77390001` may change in future iterations of the `ERC7739` standard.

But in Nexus, as well as in K1Validator (constant inherited from ERC7739Validator.sol), a constant value `0x77390001` is employed which indicates support only for the current version of the standard.

Since `SUPPORTS_ERC7739` is declared as a constant, it can break `ERC7739` flows if in future external contracts demand support for later versions of the `ERC7739` standard. There is no clarity on how different versions of the standard have to be simultaneously supported by a smart account (and by extension their validators).

Recommendation: Seek clarity on how different versions of the `ERC7739` standard have to be supported simultaneously and make a decision on how to handle this in Nexus and K1Validator.

Biconomy: Fixed in PR 216.

Cantina Managed: Fixed.

3.4.3 Nexus deviates from `ERC1271` reference implementation

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: `K1Validator :: isValidSignatureWithSender()` returns `0xffffffff` when s value is invalid while erc1271 reference implementation reverts: this can create problems when integrated with code that expects what the EIP says. This should be changed to revert for consistency across ERC1271 callers.

Recommendation: In the mentioned case, revert instead of returning `0xffffffff`.

Biconomy: Fixed in commit e7308fff.

Cantina Managed: Fixed.