0xCommit

Security Assessment Report

Biconomy

Session Key Manager V2

Version: Final

Date: 29 Feb 2024

Table of Contents

License

Disclaimer

THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHOR AND HIS EMPLOYER DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT.

Introduction

Purpose of this report

0xCommit has been engaged by **Biconomy** to perform a security audit of several Smart Account components.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behaviour.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebases Submitted for the Audit

The audit has been performed on the following commits:

Github Link:

https://github.com/bcnmy/scw-contracts/blob/feat/sma-392-session-keys-v2/contracts/smart-account/modules/SessionKeyManagers/SessionKeyManagerHybrid.sol

Version	Commit hash
Initial	f3656301f99f6d4f664cdeb304acd1624c1f5172
Final	

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
High	An attacker can successfully execute an attack that clearly results in operational issues for the service. This also includes any value loss of unclaimed funds permanently or temporary.
Medium	The service may be susceptible to an attacker carrying out an unintentional action, which could potentially disrupt its operation. Nonetheless, certain limitations exist that make it difficult for the attack to be successful.
Low	The service may be vulnerable to an attacker executing an unintended action, but the impact of the action is negligible or the likelihood of the attack succeeding is very low and there is no loss of value.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary

The status of an issue can be one of the following: **Pending**, **Acknowledged**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Overview

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line by line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
    d. Access Control Issues
    e. Boundary Analysis
4. Report preparation

Functionality Overview

The new SessionKeyManagerHybrid Module enhances Session Keys infrastructure by enabling on-chain storage of SessionData, allowing direct referencing via sessionDigest for reduced calldata and cheaper L2 transactions. It supports batched session operations for improved efficiency and transparency, and natively integrates multi-chain sessions and densely packed signatures for additional gas savings.

Summary of Findings

Sr. No.	Description	Severity	Status
1	Use of unchecked increment for loop counter	Informational	Acknowledged

Detailed Findings

1. Use of unchecked increment for loop counter.

Severity: Informational

Description

In Solidity version 0.8.22 onwards, the compiler optimizes loops by automatically using unchecked for loop counters. So, we don't need to explicitly use unchecked for loop counters.

_verifySessionEnableDataSignatures function

```
441         for (uint256 i = 0; i < length; ) {
442             _verifySessionEnableDataSignature(
443                 sessionEnableDataList[i],
444                 sessionEnableSignatureList[i],
445                 userOpSender
446             );
447             unchecked {
448                 ++i;
449             }
450         }
```

Remediation:

Since compiler version ^0.8.23 is being used, unchecked blocks for the loop counter in _verifySessionEnableDataSignatures function can safely be removed.

Status

Acknowledged