0xCommit

Security Assessment Report

Biconomy - Smart Account

Account Recovery Module

Version: Final

Date: 29 Dec 2023

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:

Version	Commit hash	Github link
Initial	fa6e294f5e233181abf787586a6d918ce5b6eb3a	https://github.com/bcnmy/scw-contracts/blob/develop/contracts/smart-account/modules/AccountRecoveryModule.sol

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 Account Recovery Module, developed by Biconomy for their Smart Contract accounts, enables the recovery of account ownership in cases where the account owner has lost their private keys. This module operates based on the concept of trusted guardians, who have the ability to sign a recovery transaction whenever necessary in order to regain access to the account.

Summary of Findings

Sr. No.	Description	Severity	Status
1	No check for security delay when initializing the module and adding guardians	Low	Resolved
2	changeGuardianParams function can add new guardians	Low	Resolved
3	Lack of kill switch can lead to abuse of account recovery module by guardians	Informatio...	Resolved
4	Use of Merkle trees will optimise operations	Informatio...	Acknowledged
5	Room for privacy	Informatio...	Acknowledged

Detailed Findings

1. No check for security delay when initializing the module and adding guardians

Severity: Low

Description

When adding a new guardian, it is ensured that *validAfter* is greater than *block.timestamp + securityDelay*. However, in the **initForSmartAccount** function, this check is not performed

Remediation

Consider adding the _checkAndAdjustValidUntilValidAfter to validate and set the time frames in the initForSmartAccount function.

Status

Resolved

Comments from Biconomy :
- It was made intentionally to allow the module to be immediately usable after initialization.

- And in case, when a user is tricked into init-ing the module, the attacker can set securityDelay = 0 (in the parameters) and the whole check will make no sense.

- The issue of tricking users into enabling/initing some module is out of the scope of the exact module, but it is more to the scope of the Smart Account itself. So the according measures to lower the potential impact of user unintentionally enabling/inits the module (for example, a security delay for all the newly enabled modules) could be introduced in the Smart Account, not in the module.

So adding the check is not required.

2. changeGuardianParams function can add new guardians

Severity: Low

Description

Whenever adding, removing, or replacing a new guardian, it is essential to check whether the guardian has already been set or not. However, in the "changeGuardianParams" function, there is no such validation. This implies that Smart accounts can utilize this function to add guardians as well.

Impact : In an ideal scenario, the owner of the smart contract should refrain from utilizing this function to add a new guardian. However, if they decide to do so, it's crucial to note that the parameter "guardiansCount" will not be incremented for that particular smart account.

Remediation

Add a check where if _guardians[guardian][msg.sender].validUntil == 0 then revert.

Status

Resolved

3. Lack of kill switch can lead to abuse of account by guardians

Severity: Informational

Description

The Account Recovery module operates by assigning guardians to facilitate administrative functions within a specified time frame. However, a potential vulnerability arises from the absence of user-defined restrictions on how frequently guardians can access the account during the recovery process. This lack of specificity opens the door to potential abuse, as guardians may exploit their access multiple times (i.e. Rogue guardian/s case) .

The optimal functionality of the recovery module should guarantee the retrieval of the account and subsequently access for guardians for a limited number of instances. For instance, in traditional Web2 services, backup codes serve as a representation of the recovery module, and once used, they regenerate for added security.

Remediation

To address this issue, a recommended remediation involves deleting the user's _smartAccountSetting post successful verification of recovery by guardians. Also to accommodate scenarios where multiple guardian accesses are necessary, the implementation of a variable, such as "recoveryCounter" in the SaSetting struct, is proposed. This counter would allow a user-defined number of recovery requests, after which all recovery settings would be automatically erased. Subsequently, the user would be required to reset or reassign guardians to ensure a secure and controlled recovery process.

Status

Resolved

Biconomy added a counter 'recoveriesLeft' to only allow a specified number of recoveries for the set of guardians.

4. Use of merkle trees will optimise operations

Severity: Informational

Description

In the existing setup, the management of guardian information for a user's account relies on a mapping structure, posing challenges for users due to the lengthy process of reverse fetching guardians from this mapping. To alleviate this complexity, leveraging a Merkle root can streamline the handling of recovery operations for users. By condensing all the necessary verification parameters into a Merkle root during setup, the process becomes more efficient.

During validation, a dynamic Merkle tree is generated on the fly, comparing its structure against the stored Merkle root. This approach not only simplifies the user experience but also optimises storage space, offering a more streamlined and resource-efficient solution for handling recovery operations.

Status

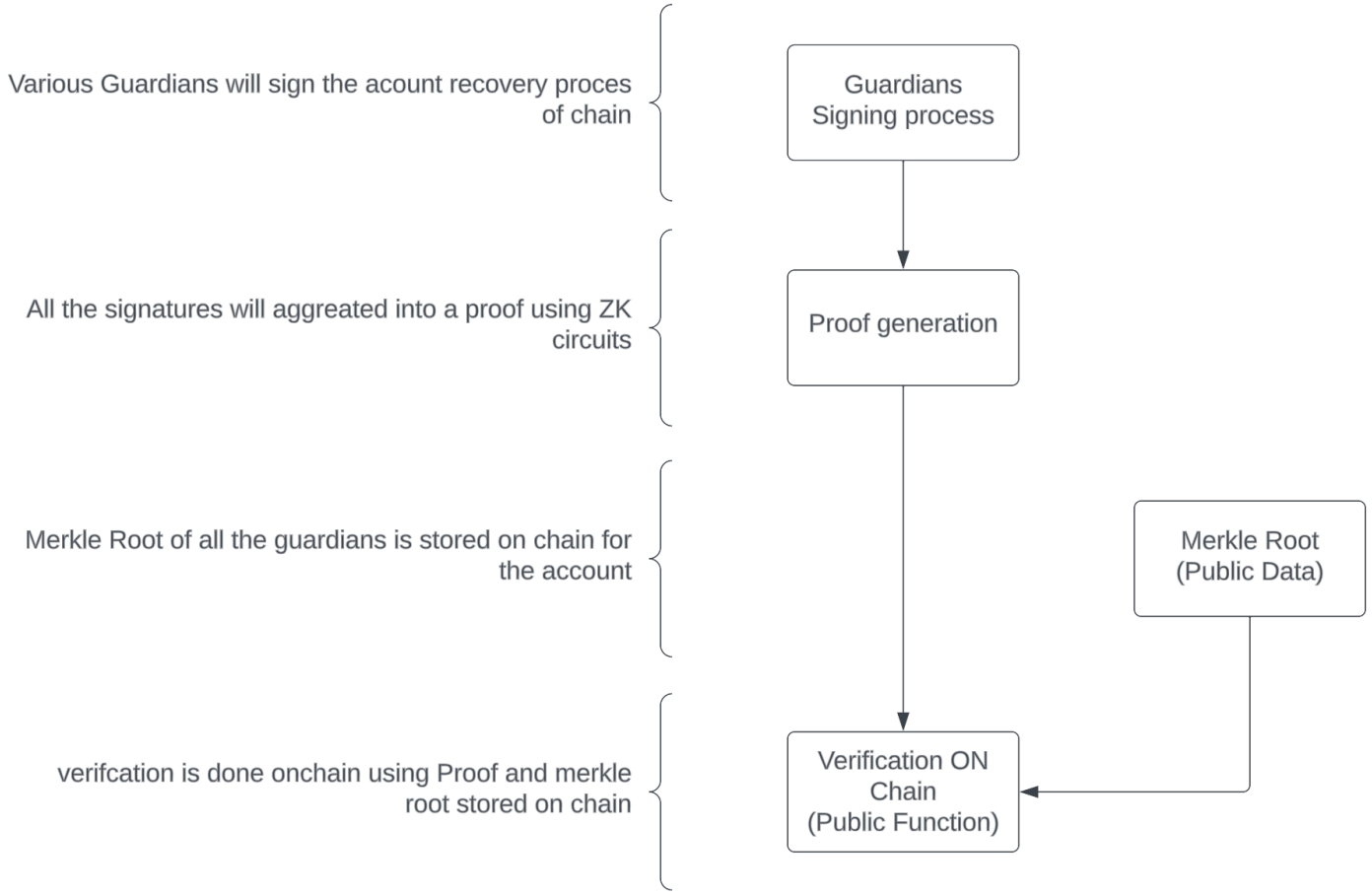
Acknowledged

5. Room for privacy

Severity: Informational

Description

Account recovery is a delicate procedure, and some users may prefer not to disclose the identities of their guardians. To address this privacy concern during recovery operations, a Zero Knowledge (ZK) approach can be employed. In this method, only the user's merkle root is stored in the recovery module. For verification, a ZK proof is provided, utilising ZK circuits to validate the signatures of guardians. This ZK proof can be verified on the blockchain using the stored merkle root, ensuring the integrity of the recovery process while maintaining the confidentiality of guardian identities. Operationally circom can be of help here.



Overview of ZK Verification process

References -

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9446757

https://eprint.iacr.org/2021/567.pdf

Status

Acknowledged