0xCommit

Security Assessment Report

Biconomy - MultiOwnedECDSA Module

Version: Final

Date: 12 Jan 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 GitHub repositories:

Version	Commit hash	Github link
Initial	dc3d1222bff6ffacd0861253e814bb2ea7266760	https://github.com/bcnmy/scw-contracts/blob/develop/contracts/smart-account/modules/MultiOwnedECDSAModule.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.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
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. This category may also include opinionated recommendations that the project team might not share.

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
4. Report preparation

Functionality Overview

The MultiOwnedECDSA Module, developed by Biconomy for their Smart Contract accounts, enables ownership of smart accounts for multiple owners. This module enables ownership and access of accounts by multiple ECDSA keys.

Summary of Findings

Sr. No.	Description	Severity	Status
1	Missing EOA Checks	Medium	Acknowledged
2	Loss of track of owners	Informatio...	Resolved
3	Unnecessary unchecked counter increment	Informatio...	Resolved
4	Unnecessary zero address check for existing owner	Informatio...	Resolved

Detailed Findings

1. Missing EOA Check

Severity: Medium

Description

The all owners added for smart accounts must be EOA accounts, but the function "initForSmartAccount" does not check if all owners added during initialization are EOA accounts.

Remediation

Add a require statement in the for loop for initForSmartAccount function which checks if the newly added owner is a EOA account or not. Following is the updated code.

```
function initForSmartAccount(
    address[] calldata eoaOwners
) external returns (address) {
    if (numberOfOwners[msg.sender] != 0) {
        revert AlreadyInitedForSmartAccount(msg.sender);
    }
    uint256 ownersToAdd = eoaOwners.length;
    for (uint256 i; i < ownersToAdd; ) {
        if (_isSmartContract(eoaOwners[i])) revert NotEOA(eoaOwners[i]);  // <- Added
        if (eoaOwners[i] == address(0))
            revert ZeroAddressNotAllowedAsOwner();
        if (_smartAccountOwners[eoaOwners[i]][msg.sender])
            revert OwnerAlreadyUsedForSmartAccount(
                eoaOwners[i],
                msg.sender
            );
        _smartAccountOwners[eoaOwners[i]][msg.sender] = true;
        unchecked {
            ++i;
        }
    }
    numberOfOwners[msg.sender] = ownersToAdd;
    return address(this);
}
```

Status

Acknowledged

2. Loss of track of owners

Severity: Informational

Description

Event “OwnershipTransferred” is used to track changes to ownership to a smart wallet. But when function “initForSmartAccount” and “addOwner” is used the OwnershipTransferred event is not triggered leading to loss of track of owners for a smart wallet. This makes it really hard to know how many actual owners are there for the account.

Remediation

Add Event “OwnershipTransferred” in “initForSmartAccount” and “addOwner” functions such that whenever an owner is added to account a event is generated and by virtue of logs list of owners for contract is derivable.

Status

Resolved

3. Unnecessary unchecked counter increment

Severity: Informational

Description

When using solidity compiler version >= 0.8.22, the compiler automatically implements the unchecked counter increment for loops. So there is no need to use explicit unchecked { ++i } in loops.

Remediation

Remove the unchecked counter increment to improve code readability.

Status

Resolved

4. Unnecessary zero address check for existing owner

Severity: Informational

Description

Whenever an owner is added zero account check is done, so when ownership is transferred there is no need to do zero account check for existing owner.

Remediation

Remove the zero account check for existing owner in the “transferOwnership” function.

Status

Resolved