0xCommit

Security Assessment Report

Biconomy - Aggregated Audit

Version: Final

Date: 12 Feb 2024

Table of Contents

License

Disclaimer

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:

Github Link: https://github.com/bcnmy/scw-contracts

Version	Commit hash
Initial	224ef5c71776d9d41aece77c23017363ab25c0c3
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.
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 Biconomy SDK serves as an Account Abstraction toolkit, streamlining user experience for dApps, wallets, and appchains. Leveraging the ERC-4337 solution, it provides a comprehensive solution to access the capabilities of the Smart Accounts Platform, Paymasters, and Bundlers.

Summary of Findings

Sr. No.	Description	Severity	Status
1	Signature length is not checked	Low	Acknowle...
2	Unused contract and modifier	Low	Resolved
3	Missing EOA checks during initialisation	Low	Acknowle...
4	Renounce Ownership can make Orphan account	Informational	Acknowle...
5	Type Mismatch in Event Emission	Informational	Resolved
6	Unused Events and errors	Informational	Resolved
7	Function isSmartContract can be converted to library function	Informational	Acknowle...
8	Possible optimisation in validation process	Informational	Resolved
9	Function isSmartContract can be converted to library function	Informational	Resolved

Detailed Findings

1. Signature length is not checked

Severity: Low

Description

In Signature decoder contract, function _splitSignature does not check for length of signature. Leading to miscalculated decoded signature.

Remediation

Add a require statement in the _splitSignature function which only executes if bytes of signature is equal to 65.

Status

Acknowledged

2. Unused contract and modifier

Severity: Low

Description

The contract base/FallbackManager and base/moduleManager inherits common/selfauthorized contract but does not use modifier authorized from common/selfauthorized contract. However in test versions of FallbackManager modifier authorized is used and has some applicability. It looks like use of modifier is required in code but not included

Remediation

If there is functional need of selfAuthorised required then there is no need for inheritance of the selfAuthorized contract and if there is need do include the use of authorized in the fallbackmanager and moduleManager contracts.

Status

Resolved

3. Missing EOA checks during initialisation

Severity: Low

Description

In EcdsaOwnershipregistry Module and MultiOwnedECDSA Module, all the 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 initForSmartAccount function which checks if the newly added owner is a EOA account or not.

Status

Acknowledged

4. Renounce Ownership can make Orphan account

Severity: **Informational**

Description

In EcdsaOwnershipregistry Module , renouncing ownership can lead to orphaning of a smart account. Such that account will be inaccessible to use.

Remediation

Ensure there is a check such that renounce ownership can not happen if there are no other modules in place.

Status

Acknowledged

5. Type Mismatch in Event Emission

Severity: Informational

Description

In Solidity, events are used to log transactions and changes in the contract state. They are crucial for off-chain applications to track contract activities. A mismatch in data types between the event definition and the event emission can lead to readability issues.

Remediation

Ensure consistent data types between event declarations and their emissions. If a uint48 is required, either the event should be declared with uint24, or the emitting value should be of type uint48.

AccountRecoveryModule.sol

```
function setSecurityDelay(uint24 newSecurityDelay) external {

        _smartAccountSettings[msg.sender].securityDelay = newSecurityDelay;

        emit SecurityDelayChanged(msg.sender, newSecurityDelay);

    }
```

Status

Resolved

6. Unused Events and errors

Severity: **Informational**

Description

Given events and errors are defined but not used anywhere.

1. ISmartAccount.sol :

```
error MixedAuthFail(address caller); #at line no 33

error OwnerCannotBeZero(); #at line no 38
```

2. IModuleManager.sol :

```
error ModulesAlreadyInitialized(); #at line no 28

error ModulesSetupExecutionFailed(); #at line no 33
```

Remediation:

When events or errors are defined in Solidity code but not used within the contract, it generally won't cause issues with the contract's internal logic or behaviour. However, it is considered good practice to review and clean up your code by removing any unused elements to enhance readability and maintainability because sometimes Unused events in contracts may indicate potential issues in the code.

Status

Resolved

7. Abstract contract for enums is unnecessary

Severity: **Informational**

Description

In base/Executor contract Enums contract is inherited. This contract only stores enums. Since enums are complex data types, they don’t need a dedicated contract.

Remediation

Define Enums without the contract.

Status

Acknowledged

8. Possible optimisation in validation process

Severity: **Informational**

Description

In ERC7484SecurityPlugin during setup of configuration there is no validation, But during access there is validation. This method will consume more gas as setup is done a limited number of times while access of configuration is done multiple times. So it's ideal to put validation checks during setup to optimise on gas consumption.

Remediation

Add condition check during setup and limited check will be needed in operations. Following is the updated codebase.

```
/// @inheritdoc IERC7484SecurityPolicyPlugin
    function setConfiguration(
        Configuration calldata _config
    ) external override {
        if (
            _config.threshold == 0 ||
            _config.trustedAttesters.length == 0
        ) {
            revert(); // Define error
        }
        _configuration[msg.sender] = _config;
        emit ConfigurationSet(msg.sender, _config);
    }

    /// @inheritdoc ISecurityPolicyPlugin
    function validateSecurityPolicy(
        address _sa,
        address _plugin
    ) external view override {
        Configuration storage saConfiguration = _configuration[_sa];

        if (
            saConfiguration.trustedAttesters.length == 0
        ) {
            revert SaConfigurationNotInitialized(_sa);
        }
        REGISTRY.checkN(
            _plugin,
            saConfiguration.trustedAttesters,
            saConfiguration.threshold
        );
    }
```

Status

Resolved

9. Function isSmartContract can be converted to library function

Severity: **Informational**

Description

There are multiple contracts which use a isSmartContract function. This function is re-written in multiple contracts, which may not be needed and can be converted to a library function as its a pure function.

Remediation

Create a library function for isSmartContract function and include the function in all contracts where it's used.

Status

Resolved