Zellic

Biconomy Multichain Validator

Smart Contract Security Assessment

August 24, 2023

Prepared for:

Ahmed Al-Balaghi

Biconomy Labs

Prepared by:

Ulrich Myhre and Yuhang Wu

Zellic Inc.

Contents

About Zellic

Zellic was founded in 2020 by a team of blockchain specialists with more than a decade of combined industry experience. We are leading experts in smart contracts and Web3 development, cryptography, web security, and reverse engineering. Before Zellic, we founded perfect blue, the top competitive hacking team in the world. Since then, our team has won countless cybersecurity contests and blockchain security events.

Zellic aims to treat clients on a case-by-case basis and to consider their individual, unique concerns and business needs. Our goal is to see the long-term success of our partners rather than simply provide a list of present security issues. Similarly, we strive to adapt to our partners' timelines and to be as available as possible. To keep up with our latest endeavors and research, check out our website zellic.io or follow @zellic_io on Twitter. If you are interested in partnering with Zellic, please contact us at hello@zellic.io.

1 Executive Summary

Zellic conducted a security assessment for Biconomy Labs from August 21st to August 22nd, 2023. During this engagement, Zellic reviewed Biconomy Multichain Validator's code for security vulnerabilities, design issues, and general weaknesses in security posture.

1.1 Goals of the Assessment

In a security assessment, goals are framed in terms of questions that we wish to answer. These questions are agreed upon through close communication between Zellic and the client. In this assessment, we sought to answer the following questions:

- Is a `userOp` that is not included in the Merkle tree validated?
- Can the `validateUserOp` function be bypassed?
- Is there a possibility of a replay attack on the `validateUserOp` function?

1.2 Non-goals and Limitations

We did not assess the following areas that were outside the scope of this engagement:

- Front-end components
- Infrastructure relating to the project
- Key custody

Due to the time-boxed nature of security assessments in general, there are limitations in the coverage an assessment can provide.

During this assessment, the lack of a comprehensive design to differentiate between multichain and single chain (currently distinguished by length) prevented us from evaluating the validity of this differentiation method.

1.3 Results

During our assessment on the scoped Biconomy Multichain Validator contracts, there were no security vulnerabilities discovered.

Zellic recorded its notes and observations from the assessment for Biconomy Labs's benefit in the Discussion section (3) at the end of the document.

2 Introduction

2.1 About Biconomy Multichain Validator

Biconomy's Multichain Validator module enables use cases that require several actions to be authorized for several chains with just one signature required from the user.

2.2 Methodology

During a security assessment, Zellic works through standard phases of security auditing including both automated testing and manual review. These processes can vary significantly per engagement, but the majority of the time is spent on a thorough manual review of the entire scope.

Alongside a variety of tools and analyzers used on an as-needed basis, Zellic focuses primarily on the following classes of security and reliability issues:

Basic coding mistakes. Many critical vulnerabilities in the past have been caused by simple, surface-level mistakes that could have easily been caught ahead of time by code review. Depending on the engagement, we may also employ sophisticated analyzers such as model checkers, theorem provers, fuzzers, and so on as necessary. We also perform a cursory review of the code to familiarize ourselves with the contracts.

Business logic errors. Business logic is the heart of any smart contract application. We examine the specifications and designs for inconsistencies, flaws, and weaknesses that create opportunities for abuse. For example, these include problems like unrealistic tokenomics or dangerous arbitrage opportunities. To the best of our abilities, time permitting, we also review the contract logic to ensure that the code implements the expected functionality as specified in the platform's design documents.

Integration risks. Several well-known exploits have not been the result of any bug within the contract itself; rather, they are an unintended consequence of the contract's interaction with the broader DeFi ecosystem. Time permitting, we review external interactions and summarize the associated risks: for example, flash loan attacks, oracle price manipulation, MEV/sandwich attacks, and so on.

Code maturity. We look for potential improvements in the codebase in general. We look for violations of industry best practices and guidelines and code quality standards. We also provide suggestions for possible optimizations, such as gas optimization, upgradeability weaknesses, centralization risks, and so on.

For each finding, Zellic assigns it an impact rating based on its severity and likelihood. There is no hard-and-fast formula for calculating a finding's impact. Instead, we assign it on a case-by-case basis based on our judgment and experience. Both the severity and likelihood of an issue affect its impact. For instance, a highly severe issue's impact may be attenuated by a low likelihood. We assign the following impact ratings (ordered by importance): Critical, High, Medium, Low, and Informational.

Zellic organizes its reports such that the most important findings come first in the document, rather than being strictly ordered on impact alone. Thus, we may sometimes emphasize an "Informational" finding higher than a "Low" finding. The key distinction is that although certain findings may have the same impact rating, their *importance* may differ. This varies based on various soft factors, like our clients' threat models, their business needs, and so on. We aim to provide useful and actionable advice to our partners considering their long-term goals, rather than a simple list of security issues at present.

Finally, Zellic provides a list of miscellaneous observations that do not have security impact or are not directly related to the scoped contracts itself. These observations — found in the Discussion (3) section of the document — may include suggestions for improving the codebase, or general recommendations, but do not necessarily convey that we suggest a code change.

2.3 Scope

The engagement involved a review of the following targets:

Biconomy Multichain Validator Contracts

Repository	https://github.com/bcnmy/scw-contracts
Version	scw-contracts: `b8d4e08ebe27b721bc746b430f29f41791144549`
Program	MultichainECDSAValidator
Type	Solidity
Platform	EVM-compatible

2.4 Project Overview

Zellic was contracted to perform a security assessment with two consultants for a total of four person-days. The assessment was conducted over the course of two calendar days.

Contact Information

The following project manager was associated with the engagement:

Chad McDonald, Engagement Manager
chad@zellic.io

The following consultants were engaged to conduct the assessment:

Ulrich Myhre, Engineer
unblvr@zellic.io

Yuhang Wu, Engineer
yuhang@zellic.io

2.5 Project Timeline

The key dates of the engagement are detailed below.

August 21, 2023	Kick-off call
August 22, 2023	Start of primary review period
August 23, 2023	End of primary review period

3 Discussion

The purpose of this section is to document miscellaneous observations that we made during the assessment. These discussion notes are not necessarily security related and do not convey that we are suggesting a code change.

3.1 Chosen validation mode lacks explicity

When a user of the Smart Account wants to authorize several actions for several chains, and only use one signature to do so, they set the "chosen authorization module address" to the address of a deployed MultichainECDSAValidator module. The module function `validateUserOp()` will call `_verifySignature()` with either the `userOpHash` or the extracted `merkleTreeRoot` as part of the input parameters. The decision for which parameters to pick is not explicitly chosen by the caller, but instead relies on the length of the signature data being exactly 65 or not. We believe that showing the intention more explicitly could be safer. If a Merkle signature is specially picked, it is possible to make it go to the non-Merkle code path, where unexpected return values, events, or errors could arise.

Explicit intentions also slightly future-proof the module. If the module is likely to be expanded further, a third differentiator will be required. If this new type cannot be differentiated by length alone, the module will need refactoring to include a new field or extra data. That breaks compatibility with existing contracts that interact with the module, which requires updates for users.

There is no vulnerability related to the current code pattern, but future expansions of the project can easily convert this implicit check into a dangerous mistake or break backward compatibility. A more explicit solution could be to include a signature type inside the signature itself.

This issue has been acknowledged by Biconomy Labs.

Biconomy Labs stated,

> We've decided to still go with the current approach without explicitly adding singlechain/multichain mode flag into some userOp data. Usually multichain mode has 65 bytes for ecdsa signature over the merkle root + some additional info (validUntil, validAfter, merkleRoot, merkleProof) so it will never be 65 bytes, but always more.

4 Threat Model

This provides a full threat model description for various functions. As time permitted, we analyzed each function in the contracts and created a written threat model for some critical functions. A threat model documents a given function's externally controllable inputs and how an attacker could leverage each input to cause harm.

Not all functions in the audit scope may have been modeled. The absence of a threat model in this section does not necessarily suggest that a function is safe.

4.1 Module: MultichainECDSAValidator.sol

Function: `validateUserOp(UserOperation userOp, byte[32] userOpHash)`

This validates user operation.

Inputs

- `userOp`
  - **Control**: Could be controlled by the user.
  - **Constraints**: The data should be a valid `UserOperation`.
  - **Impact**: The user operation to be validated.
- `userOpHash`
  - **Control**: Could be controlled by the user.
  - **Constraints**: N/A.
  - **Impact**: The hash of the `userOp` provided by the EP.

Branches and code coverage (including function calls)

Intended branches

- Test result is correct if it is a single-chain signature.
  ☑ Test coverage
- Test result is correct if it is a multichain signature.
  ☑ Test coverage

Negative behavior

- Validation failed due to signature error.
  ☑ Negative test
- Validation reverted due to invalid `UserOp`.

- [x] Negative test

5 Assessment Results

At the time of our assessment, the reviewed code was not deployed to the Ethereum Mainnet.

During our assessment on the scoped Biconomy Multichain Validator contracts, there were no security vulnerabilities discovered.

5.1 Disclaimer

This assessment does not provide any warranties about finding all possible issues within its scope; in other words, the evaluation results do not guarantee the absence of any subsequent issues. Zellic, of course, also cannot make guarantees about any code added to the project after the version reviewed during our assessment. Furthermore, because a single assessment can never be considered comprehensive, we always recommend multiple independent assessments paired with a bug bounty program.

For each finding, Zellic provides a recommended solution. All code samples in these recommendations are intended to convey how an issue may be resolved (i.e., the idea), but they may not be tested or functional code. These recommendations are not exhaustive, and we encourage our partners to consider them as a starting point for further discussion. We are happy to provide additional guidance and advice as needed.

Finally, the contents of this assessment report are for informational purposes only; do not construe any information in this report as legal, tax, investment, or financial advice. Nothing contained in this report constitutes a solicitation or endorsement of a project by Zellic.