SPEARBIT

ZkSync snark-wrapper Security Review

zkSync snark-wrapper, era-boojum and rescue-poseidon security review

Auditors

Wilson Nguyen, Lead Security Researcher

Nirvan, Lead Security Researcher

Report prepared by: Lucas Goiriz

November 28, 2023

Contents

1 About Spearbit

Spearbit is a decentralized network of expert security engineers offering reviews and other security related services to Web3 projects with the goal of creating a stronger ecosystem. Our network has experience on every part of the blockchain technology stack, including but not limited to protocol design, smart contracts and the Solidity compiler. Spearbit brings in untapped security talent by enabling expert freelance auditors seeking flexibility to work on interesting projects together.

Learn more about us at spearbit.com

2 Introduction

zkSync is the layer 2 protocol that scales Ethereum's security and values through zero-knowledge cryptography. The mission of zkSync is to accelerate the mass adoption of crypto for personal sovereignty.

Disclaimer: This security review does not guarantee against a hack. It is a snapshot in time of `snark-wrapper`, `era-boojum` and `rescue-poseidon` according to the specific commit. Any modifications to the code will require a new security review.

3 Risk classification

Severity level	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

3.1 Impact

- High - leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
- Medium - global losses <10% or losses to only a subset of users, but still unacceptable.
- Low - losses will be annoying but bearable--applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.

3.2 Likelihood

- High - almost certain to happen, easy to perform, or not easy but highly incentivized
- Medium - only conditionally possible or incentivized, but still relatively likely
- Low - requires stars to align, or little-to-no incentive

3.3 Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

4 Executive Summary

Over the course of 14 days in total, ZkSync engaged with Spearbit to review snark-wrapper, era-boojum and rescue-poseidon protocols. The codebase in scope consisted in the constraints proved by a PLONK-like SNARK during the final compression step of the zkSync pipeline. These constraints encode a verifier for the Boojum STARK (a Plonky2-like STARK which supports lookup arguments). The goal of the review was to verify that the constraints correctly encode the native Boojum STARK verifier (the purpose of the review was not to verify the soundness of the STARK, SNARK or overall proof system).

The constraints translation mapping is summarized by the below table:

Functionality	**Native**	**Constraints**
Poseidon2 Round Function	`rescue-poseidon/src/poseidon2/poseidon2.rs`	`rescue-poseidon/src/circuit/{poseidon2.rs, sbox.rs, matrix.rs}`
Poseidon2 Sponge	`rescue-poseidon/src/poseidon2/sponge.rs`	`rescue-poseidon/src/implementations/poseidon2/mod.rs`
Poseidon2 TreeHasher	`rescue-poseidon/src/poseidon2/sponge.rs`	`snark-wrapper/src/implementations/poseidon2/tree_hasher.rs`
Poseidon2 Transcript	`rescue-poseidon/src/poseidon2/transcript.rs`	`snark-wrapper/src/implementations/poseidon2/transcript.rs`
Verifier	`era-boojum/src/cs/implementations/verifier.rs`	`snark-wrapper/src/verifier/{mod.rs, first_step.rs, quotient_contributions.rs, fri.rs}`

During the review, no translation errors between the constraints and native Boojum verifier were found. In this period of time a total of **2** issues were found.

Summary

Project Name	ZkSync
Repositories	snark-wrapper, era-boojum, rescue-poseidon
Commits	52f9ef98, 84754b06, c4a78847
Type of Project	zkEVM
Audit Timeline	Oct 20 to Nov 3

Issues Found

Severity	**Count**	**Fixed**	**Acknowledged**
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	2	0	0
Optimizations	0	0	0
Informational	0	0	0
Total	**2**	**0**	**0**

5 Findings

5.1 Low Risk

5.1.1 Public Inputs Encoding Unused

Context: snark-wrapper/mod.rs#L136

Description: The `aggregate_public_inputs` function generates a constraint system variable `pi` that is constrained to be a BN256 field element which is the encoding of the Boojum verifier's public input (an array of Goldilocks field elements). Our concern is that this variable `pi` is effectively unused besides these initial constraints. In particular, during the constraint system synthesis (i.e. the `synthesize` function), the output of the call to `aggregate_public_inputs` is unused. We reached out to the zkSync team and they state that this variable is meant to expose the public input to the L1 verifier.

Recommendation: Expose the aggregated BN256 element as public input by allocating an aggregated public input variable and enforcing equality to the computed aggregate.

5.1.2 Encoding of Goldilocks Field Elements

Context: snark-wrapper/mod.rs#L207, era-zkevm_circuits/mod.rs

Description: In `aggregate_public_inputs`, there is an explicit assumption on the width of the Goldilocks field elements which represent the public input.

```
let chunk_bit_size = (GL::CAPACITY_BITS / 8) * 8; assert!(public_inputs.len() * chunk_bit_size <=
↪  E::Fr::CAPACITY as usize, "scalar field capacity is not enough to fit all public inputs");
```

In particular, `chunk_bit_size` is set to 56 bits (7 bytes). The algorithm accumulates an array of Goldilocks field elements into a single BN256 field element by performing a rolling sum. In each step of the sum, the prior sum is shifted by 56 bits to make room for a new Goldilocks element to be accumulated. An arbitrary Goldilocks field element is represented by 64 bits (for example, `p-2` requires all 64 bits to be represented). This leads to a potential encoding issue.

During the encoding process, a Goldilocks field element which is greater than 56 bits could be used to overwrite bits of earlier public input elements. We reached out to the zkSync team and they state that the public input elements are restricted to 56 bits by a `scheduler` circuit proven during their pipeline. We wanted to note these encoding concerns explicitly, just in case future use of the verifier constraints is not done in tandem with the `scheduler` circuit.

Recommendation: Explicitly enforce 56-bit range constraints on the input field elements.