SALUS SECURITY

FEB 2024

CODE SECURITY ASSESSMENT

ZKSHUFFLE

Overview

Project Summary

- Name: zkShuffle
- Language: Circom
- Repository:
    - https://github.com/Manta-Network/zkShuffle
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	zkShuffle
Version	v2
Type	Circom
Dates	Feb 1 2024
Logs	Jan 31 2024, Feb 1 2024

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	1
Total informational issues	0
Total	1

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue can lead to substantial financial, reputation, availability, or privacy damage.
Medium Risk	The issue can lead to moderate financial, reputation, availability, or privacy damage. Or the issue can lead to substantial damage under extreme and unlikely circumstances.
Low Risk	The issue does not pose an immediate security threat, but may be a lack of following best practices or more easily lead to the future introductions of bugs.
Informational	Information not relevant to security, but may be helpful for efficiency, costs, etc.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Architectural Design
- Logic Error
- Underconstrained Signal
- Information Leakage
- Access Control
- Data Validation
- Overflow/Underflow
- Bad Randomness
- Denial of Service
- Redundancy

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s), circuit(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Underconstrained inputs in ecDecompress	Low	Underconstrained Signal	Acknowledged

2.2 Detailed Findings

1. Underconstrained inputs in ecDecompress

Severity: Low	Category: Underconstrained Signal
Target: - packages/circuits/circuits/common/babyjubjub.circom	

Description

The **ecDecompress** template uses an input signal s to assign a positive or negative sign to delta (note that negative numbers are in finite field). The following snippet shows where the issue lies in:

babyjubjub.circom: L6-L34

```
template ecDecompress() {
...
    tmp[0] <== s*delta;
    tmp[1] <== (s-1) * delta;
    y <== tmp[0] + tmp[1];
...
}
```

The s signal is a boolean selector but without boolean constraints. Although the input signal s from a project global perspective is indeed a boolean value, the implementation is not conducive to template reuse.

Impact: No impact to the current code base but if anyone uses the template without constraining the input signal s to a boolean value, this will allow attackers to forge proof using arbitrary value.

Recommendation

Document the function to warn the caller that they must ensure the s input signal is well-constrained to be boolean value.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following file in commit 7278810:

File	SHA-1 hash
packages/circuits/circuits/common/babyjubjub.circom	63ea2110838c46a3a999466a2463c6a814cf1e03
packages/circuits/circuits/common/boolean.circom	b82973fb94ffb6c9da255e4246cbec4b380cdb59
packages/circuits/circuits/common/elgamal.circom	c539af0e1c09c155fd0529d5db29a5eff0a6df15
packages/circuits/circuits/common/matrix.circom	92afb40779781bfce17e2212219cf30bc4310d5e
packages/circuits/circuits/common/permutation.circom	6069300070d596b2990d30e12490ce01607bf4ac
packages/circuits/circuits/decrypt/decrypt.circom	0b8e1943f73d5d882f481cba972716f101649c1c
packages/circuits/circuits/shuffle_encrypt/shuffle_encrypt_template.circom	f8d8e50c0be77b4c69025f79519e115b66646250
packages/circuits/circuits/shuffle_encrypt/shuffle_encrypt.circom	3f2ab5939792d757c0c0786be19a7c00266d64d5