Semaphore Security Audit

By Kev Zettler, Security Engineer @ EF’s Privacy & Scaling Explorations Team

October 2022

Table of Contents

Overview

Executive Summary

This audit reviews Semaphore version 2.5.0 as a follow up audit to a previous audit of semaphore 2.0.0. This audit looks at the modifications and additions to the Semaphore codebase between those versions. This audit did not review the circom circuits as they have not changed since the previous audit. The new major changes that this audit reviews are modifications to the Semaphore smart contracts, changes to the dependencies in zk-kit, and the new Semaphore Javascript NPM modules.

Background

Semaphore is a set of tools and contracts that enable developers to add anonymous voting or whistleblowing features to their dapps. The latest version of Semaphore provides both solidity and JavaScript libraries for developers to consume.

Coverage

Github Repo: https://github.com/semaphore-protocol/semaphore
Commit Hash: 68779e90a0db120d9c36143c5f48ca6fd1a2a159
Release Tag: v2.5.0
Documentation: https://semaphore.appliedzkp.org/

The audit covers all contracts in semaphore/contracts, ZK-kit IncrementalBinaryTree and SparseMerkleTree data structures, Javascript Group, Identity, and Proof modules The following were focused on during the review of the smart contracts (but not limited to):

- Proper access control for handling group settings and membership
- Identity revealing information kept off-chain and private when proving membership
- Inability to successfully cast a vote or whistleblow when a proof is invalid
- Nullifier hashes were properly set when required
- Values were within range of the snark field when required
- Malicious members cannot compromise the privacy of others or halt the protocol

Techniques Used

The following techniques were used in this audit. Manual review was used across the Semaphore contracts, ZK-Kit dependencies, and Javascript modules. Static Analysis via Slither and Mythril was used on the Semaphore contracts.

Static Analysis - Mythril

Mythril is a static analysis tool from ConsenSys. Mythril compiles and runs against contract bytecode. Mythril reported no issues with the Semaphore contracts.

```
$ myth analyze ./contracts/Semaphore.sol --solc-json ./myth-standard-solc.json
The analysis was completed successfully. No issues were detected.
```

Static Analysis - Slither

Slither is a static analysis tool from TrailOfBits. Slither was failing to parse some of the Semaphore code. Some of the Semaphore code had to be modified to complete a slither scan. Slither struggles to parse shared library classes. IncrementalBinaryTree.sol was modified to move the struct within the library and then had to patch SemaphoreGroups.sol to update the references to IncrementalBinaryTree.IncrementalTreeData. The verifier contracts are modified to include a contract local error class for InvalidProof. These slither issues have been reported at:

https://github.com/crytic/slither/issues/1393
https://github.com/crytic/slither/issues/1379

Slither reported many results. These results were inspected further for anything high, or critical but no results were found. Full slither results available at:

https://gist.github.com/kevzettler/6e8e9174aabfa0ea271c81381e5f6b2f

General Analysis

Category	Evaluation
Access Control	**Strong.** Access is limited as intended, mainly to group admins.
Launch Risk	**Strong.** Semaphore does not manage assets. Additionally many dApps built on Semaphore will deploy their own Semaphore contract. They should consider launch controls if necessary.
Code Quality	**Strong.** Code follows best practices for solidity and circom. No unnecessary use of assembly. No confusing variable/function

	names. Good use of interfaces and inheritance.
Events	**Strong.** Events are emitted after every important function call such as casting votes and adding members. Relevant details are emitted.
Dummy Proof	**Strong.** Contract function names are clear in their intent and hard to misuse.
Complexity	**Strong. Short and simple contracts and circuits**
Testing	**Strong.** Contracts and Javascript modules have good test coverage.
Documentation	**Strong.** NatSpec comments for all functions and good documentation on the website.
Cryptography	**Not Reviewed**
ZK Circuits	**Not Reviewed**

Findings

The audit found no severe issues during manual review. The audit found varying severity issues during the slither audit that upon manual review were deemed false positives or low severity.

Major

[M1] IncrementalBinaryMerkleTree.sol: Update can change uninitialized leaves in the tree.

https://github.com/privacy-scaling-explorations/zk-kit/issues/32

During the audit an issue was filed against the zk-kit repo pointing out an error in the IncrementalBinaryMerkleTree.sol contract

IncrementalBinaryMerkleTree.sol: Update can change uninitialized leaves in the tree.

Describe the bug It is possible to use the `update` function to change a zero leaf at an index that hasn't been inserted into the tree yet.

For example, suppose that 4 leaves are inserted into the tree. We can update leaf 7 by proving inclusion of the zero value at leaf 7, because technically the tree is never "empty", but it begins completely filled with zero values.

Updating an uninitialized index can cause the tree root to no longer represent the set of leaves accurately, because the lastSubtrees array will be incorrect at some point when updating the root.

Fix Log

Issue	Severity	Status
[M1]	Major	Fixed https://github.com/privacy-scaling-explorations/zk-kit/issues/32

Vulnerability Classifications

Severity Categories	
Severity	**Description**
Recommendation	Information not relevant to security, but may be helpful for efficiency, costs, etc.
Warning	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.
Minor	The code does not work as intended. Impact to the system and users is minimal if present at all.
Major	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.
Critical	The issue can lead to **substantial** financial, reputation, availability, or privacy damage.