Tornado pool audit

1 August 2021, Igor Gulamov

Introduction

Igor Gulamov conducted the audit of tornado.cash smart contracts and circuits.

This review was performed by an independent reviewer under fixed rate.

Scope

zkSNARKs & Solidity contracts from tornado-pool, including PR with OVM support.

Issues

We found no critical or major issues.

We consider commit b085ab398eaeefff98771f5dad893cb804d98e70 as a safe version from the informational security point of view.

We consider commit 9931fdebefa6de3a0e4f8884406f593b354d3ddf as a safe version from the informational security point of view for OVM implementation.

Critical

Major

Warnings

1. merkleTree.circom#L17-L25

Unoptimized circuit. We propose rewriting it as the following:

```
out[1] <== in[0] + in[1] - out[0];
```

wont fix

2. transaction.circom#L23

From the zkSNARK side `extAmount` and `fee` means the same thing: difference between transaction input and output amount. We propose replacing it with one in-SNARK variable and storing the details inside `extData` structure.

Fix Fix2

3. transaction.circom#L36

It's enough to publish the subtree hash only. Leaves could be stored at `extData`.

wont fix

4. TornadoPool.sol#L115

The expression could be optimized as

```
    return int256(_extAmount-FIELD_SIZE);
```

Refactored in another fix

5. TornadoPool.sol#L102-L103

We propose caching `currentCommitmentIndex` for gas optimizations.

Fix

6. TornadoPool.sol#L102-L106

Event data is available from calldata. We propose replacing these events with

```
    emit NewTransaction(_currentCommitmentIndex);
```

wont fix

7. transaction.circom#L122

We recommend to implement integer log2 function here or add `assert(nOuts==2)`.

Comments

1. treeUpdater.circom#L6

We propose adding `nLeaves` template parameter here with `assert(nLeaves==2);` for improving the readability of the circuit.

Fix

2. transaction.circom#L23

Wrong sign in the description. We propose fixing it

```
// publicAmount = -fee + extAmount
```

Fix

Severity Terms

Comment

Comment issues are generally subjective in nature, or potentially deal with topics like "best practices" or "readability". Comment issues in general will not indicate an actual problem or bug in code.

The maintainers should use their own judgment as to whether addressing these issues improves the codebase.

Warning

Warning issues are generally objective in nature but do not represent actual bugs or security problems.

These issues should be addressed unless there is a clear reason not to.

Major

Major issues will be things like bugs or security vulnerabilities. These issues may not be directly exploitable, or may require a certain condition to arise in order to be exploited.

Left unaddressed these issues are highly likely to cause problems with the operation of the contract or lead to a situation which allows the system to be exploited in some way.

Critical

Critical issues are directly exploitable bugs or security vulnerabilities.

Left unaddressed these issues are highly likely or guaranteed to cause major problems or potentially a full failure in the operations of the contract.