TornadoCash Project Security Audit

Tornado.sol

Audit Resources:

Github Repository of the project was provided. (Link)

Project Author:

- TornadoCash community (Github)

Project Auditor:

- Umair Mirza (Github)

Table of Contents

Audit Summary

The TornadoCore project has been compiled, deployed and tested using the Truffle smart contract development tool chain. The project is comprised of two smart contracts, namely:

- Tornado.sol
- Verifier.sol
- MerkleTreeWithHistory.sol
-

Following contracts have been audited by the Auditor:

- Tornado.sol

The contract has been audited by 1 resident from September 10th October to 12th October 2022.

Scope

The scope of this audit is limited to the smart contracts mentioned below:

- Tornado.sol

The commit that has been audited is: **1ef6a263ac6a0e476d063fcb269a9df65a1bd56a**
This audit is about identifying potential vulnerabilities in the smart contracts. The audit may not identify all potential attack vectors or areas of vulnerability.

Code Evaluation Matrix

Category	Mark	Description
Access Control	Good	Access control is appropriate for this contract
Compiler	Medium	Solidity 0.7.0 and Solc 0.7.0 are used which are old and do not use the latest security checks
Complexity	Medium	Similar variable names create a bit of confusion while reviewing. Also the child contract function looks confusing at first. Child contract names are not mentioned.
Libraries	Good	Openzeppelin is being used for Reentrancy Guard which is a good solution to prevent Reentrancy.

Decentralisation	Good	The contract follows the rules of decentralisation.
Code Stability	Good	The last commit to the repository was in March 2022. The code has not been modified since then.
Documentation	Average	Comments exist in many places but the contract are missing the recommended Natspec or similar specification.
Monitoring	Good	Events have been added to all important functions in the contract
Testing & Verification	Good	All tests are passing.

Findings Description

Findings have been broken down into sections by their respective impact:

- Critical, High, Medium, Low Impact
    - These are findings that range from attacks that may cause loss of funds, impact control/ownership of the contracts, or cause any unintended consequences/actions that are outside the scope of the requirements.
- Gas Savings
    - Findings that can improve the gas efficiency of the contracts.
- Informational
    - Findings including recommendations and best practices.

Informational Findings

1. Informational - Different versions of Solidity are used

Proof of Concept

- Version used: ['>=0.6.0<0.8.0', '^0.7.0']
    - ^0.7.0 (contracts/MerkleTreeWithHistory.sol#13)
    - ^0.7.0 (contracts/Tornado.sol#13)
    - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/ReentrancyGuard.sol#3)

Impact

Different Solidity versions in the same project can cause undesired side-effects in some situations.

Recommendation

Use one Solidity version.

2. Informational - Solc 0.7.0 is not recommended for deployment

Proof of Concept

```
pragma solidity ^0.7.0;
```

Impact

Using an old version prevents access to new Solidity security checks. We also recommend avoiding complex pragma statement.

Recommendation

Deploy with any of the following Solidity versions:

- 0.5.16 - 0.5.17
- 0.6.11 - 0.6.12
- 0.7.5 - 0.7.6
- 0.8.16
- or latest versions

3. Informational - Variable names are too similar

Proof of Concept

Variable Tornado.deposit(bytes32)._commitment (contracts/Tornado.sol#55) is too similar to Tornado.commitments (contracts/Tornado.sol#28)

```
mapping(bytes32 => bool) public commitments;
function deposit(bytes32 _commitment) external payable nonReentrant {
```

Impact

If you use several variables with similar names, the contract code will be difficult to review.

Recommendation

Prevent variables from having similar names.

4. Informational - Child contract not mentioned in reference

Proof of Concept

The function _processDeposit() is defined in the Tornado.sol contract (contracts/Tornado.sol#66) but the child contract name is not mentioned.

```
  /** @dev this function is defined in a child contract */
  function _processDeposit() internal virtual;
```

Impact

Code reviewer can mistakenly look into another child contract that is not intended here.

Recommendation

The name of the child contract where the function is defined should be mentioned here.