zk2048

Audit Report

Tue Sep 05 2023

contact@scalebit.xyz https://twitter.com/scalebit_

ScaleBit

zk2048 Audit Report

1 Executive Summary

1.1 Project Information

Description	An zero-knowledge proof implentation of 2048 game
Type	Game
Auditors	ScaleBit
Timeline	Mon Aug 28 2023 - Tue Sep 05 2023
Languages	Circom
Platform	Starknet
Methods	Architecture Review, Unit Testing, Manual Review

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
204	2048.circom	7d9bbb9c2b5ced6ce59c2be85d57c3b86388b4ac

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	6	5	1
Informational	0	0	0
Minor	1	1	0
Medium	1	0	1
Major	3	3	0
Critical	1	1	0

1.4 ScaleBit Audit Breakdown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for in the field of circuits included (but are not limited to):

- Under-constrained issue.
- Non-deterministic circuit.
- Operator misuse.
- Unused signal.
- Finite field overflow/underflow
- Over-constrained issue.
- Unsafe parameter or assertion.
- Redundant constraints.
- Non-quadratic arithmetic optimization.

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

- We will using local environment to check the circuit with valid random seed and examine the output. The items to check are: proof examination / redundant constraint / parameter verification / overflow and underflow / unused signal.

(2) Code Review

- The code scope is illustrated in section 1.2.

(3) Formal Verification

- Perform formal verification for key functions by converted the circuit into SMT-lib, and leverage Z3 and CVC5 to perform model checking.

(4) Audit Process

- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);

- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

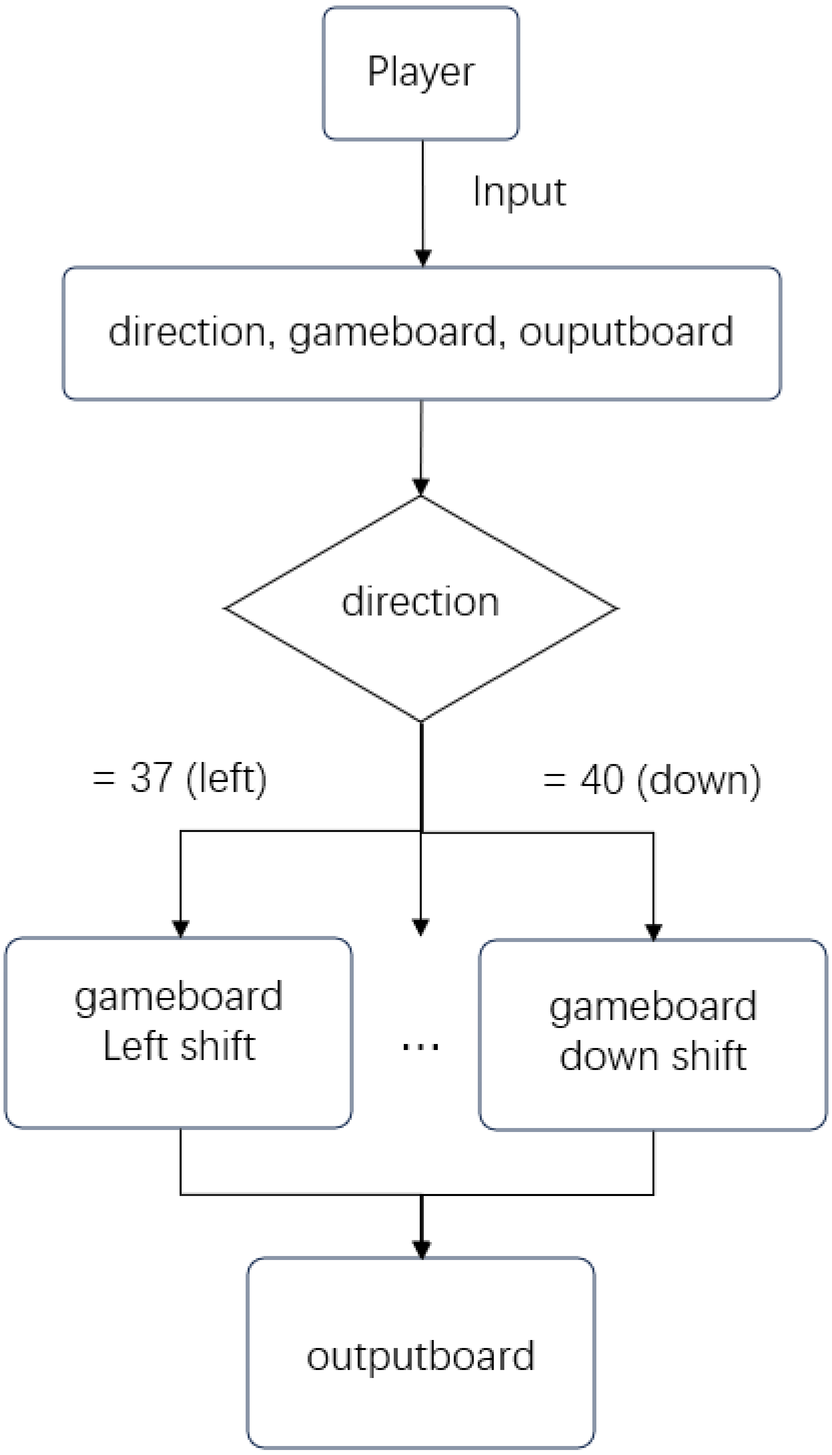
This report has been commissioned by ZKgamestop to identify any potential issues and vulnerabilities in the source code of the zk2048 smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 6 issues of varying severity, listed below.

ID	Title	Severity	Status
204-1	Potential Invaild Input	Critical	Fixed
204-2	Misuse Operator	Major	Fixed
204-3	Signal is not constrained	Major	Fixed
204-4	Arithmetic Underflow	Major	Fixed
204-5	Unexpected output	Medium	Acknowledged
204-6	Redundant Signal	Minor	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the zk2048 Smart Contract:



Notice that the final output of the circuit is the proof, but not `outputboard` or any `output signal` of the circuit.

4 Findings

204-1 Potential Invaild Input

Severity: Critical

Status: Fixed

Code Location:

2048.circom#1

Descriptions:

In the `2048` game, the input direction and the number on board are strictly limited to `up`, `down`, `left`, and `right`. However, there is no examination of the input signal `direction`.

```
signal input direction;
signal input gameboard[4][4];
signal output ouputboard[4][4];
...
if (direction == 39){
      for( var row = 0;row < 4 ;row++)
      { ...}
}
```

The lack of input assertion may lead to unexpected results. In this case, when `direction = 3`, it will bypass all of the `if` branches, and end up with no constraint added in the `gameboard`. Therefore, the attacker could forge infinite proof.

Suggestion:

Check `(direction-37)*(direction-38)*(direction-39)*(direction-40) == 0`, and the elements inside the `gameboard` should be constrained by a template that examie whether they are `2^n`.

204-2 Misuse Operator

Severity: Major

Status: Fixed

Code Location:

2048.circom#260-261

Descriptions:

Using the signal assignment operator `<--` does not constrain the assigned signal, which leads to the assigned signal `ouputboard[i][j]` not being constrained here.

```
for(var  i =0;i<4;i++){
    for(var j=0;j<4;j++ ){
        ouputboard[i][j] <-- gamedata[i][j];
    }
}
```

This issue will lead to an Under-Constrained issue and the attack may leverage it to forge fake proof.

Suggestion:

fix the operateor from `<--` to `<==`

204-3 Signal is not constrained

Severity: Major

Status: Fixed

Code Location:

2048.circom#8

Descriptions:

The input signal `ouputboard[4][4]` does not occur in a constraint. Any Under-Constrained signal may be leveraged by the attacker to forge fake proof.

Suggestion:

Try to recheck the function of this signal. If it useless, considered using a dummy constrain like `x_square === x * x`.

204-4 Arithmetic Underflow

Severity: Major

Status: Fixed

Code Location:

2048.circom#89,145

Descriptions:

In the loop `for(var i = 3; i >= 0; i--)` , the `i` will be assigned to `p-1` at the end of the loop.

```
if (direction == 39){
        for( var row = 0;row < 4 ;row++){
        for(var i =3; i >= 0;i--){
            var k=0;
            for(var j = i-1 ;j>= 0;j--){  i,j,gamedata[row][i] ,gamedata[row][j] );
                if (gamedata[row][i] == 0 && gamedata[row][j] > 0 && k ==0){
                    gamedata[row][i]= gamedata[row][j];
                    gamedata[row][j] = 0;
                    k=1;
                }
            }
        }
```

Suggestion:

Rewrite the condition in the `for` loop, and try to use `>0` to avoid underflow.

204-5 Unexpected output

Severity: Medium

Status: Acknowledged

Code Location:

2048.circom#262-266

Descriptions:

According to the rules of `2048` game, when the final output is large or equal to `2048`, the game should be over, therefore, the circuit should reject this invalid output and not generate a proof.

```
for(var i=0;i<4;i++){
    for(var j=0;j<4;j++ ){
        ouputboard[i][j] <-- gamedata[i][j];
    }
}
```

However, in the circuit, there is no constraint that ensures all the value inside `outputboard` is less than 2048.

Suggestion:

Add assertion to ensure no elements in the `outputboard` are larger than 2048, by traversal the `outputboard` and comparing each element with `2048`.

204-6 Redundant Signal

Severity: Minor

Status: Fixed

Code Location:

2048.circom#27

Descriptions:

The input signal `s` does not occur in a constraint, and it appears to be useless in the whole circuit.

```
signal output s <== gamedata[0][0] * gamedata[0][0];
```

The ineffective signal is considered an over-constrained issue, and it may cause the circuit to cost more time to compile.

Suggestion:

Consider removing this signal in the circuit.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.